





FUNDAMENTOS Y PRÁCTICAS  
DE  
BASES DE DATOS



# FUNDAMENTOS Y PRÁCTICAS DE BASES DE DATOS

Enrique Alegre Gutiérrez  
Ramón Ángel Fernández Díaz  
Lidia Sánchez González



UNIVERSIDAD DE LEÓN  
Secretariado de Publicaciones  
y Medios Audiovisuales  
2003

ALEGRE GUTIÉRREZ, Enrique

Fundamentos y prácticas de bases de datos / Enrique Alegre Gutiérrez, Ramón Ángel Fernández Díaz, Lidia Sánchez González. – León : Universidad, Secretariado de Publicaciones y Medios Audiovisuales, 2003

262 p.; 24 cm.

Bibliogr.

ISBN 84-9773-040-2

1. Bases de datos. I. Fernández Díaz, Ramón Ángel. II. Sánchez González, Lidia. III. Universidad de León. Secretariado de Publicaciones y Medios Audiovisuales. IV. Título  
681.31.07

*Queda prohibida cualquier forma de reproducción y transformación de esta obra sin la autorización de los titulares de la propiedad intelectual, lo cual puede ser constitutivo de delito (art.270 y ss. del Código Penal).*

© UNIVERSIDAD DE LEÓN  
Secretariado de Publicaciones y Medios Audiovisuales  
Los Autores

ISBN: 84-9773-040-2  
Depósito Legal: LE-978-2003

Impresión: Servicio de Imprenta. Universidad de León  
Campus de Vegazana.  
24071 LEÓN

# ÍNDICE

|   |           |
|---|-----------|
| <b>TEORÍA .....</b>                         | <b>17</b> |
| <b>TEMA 1. INTRODUCCIÓN .....</b>           | <b>19</b> |
| 1. Interés de las Bases de Datos. ....      | 19        |
| 1.1. Sin Bases de Datos. ....               | 19        |
| 1.2. Inconvenientes. ....                   | 21        |
| 1.3. Solución. ....                         | 21        |
| 2. Visión de los Datos. ....                | 21        |
| 2.1. Sistema Gestor de Bases de Datos. .... | 21        |
| 2.2. Abstracción. ....                      | 22        |
| 2.3. Ejemplar y Esquema. ....               | 22        |
| 2.4. Independencia de Datos. ....           | 22        |
| 3. Modelos de Datos. ....                   | 23        |
| 3.1. Lógicos basados en objetos. ....       | 23        |
| 3.1.1. Entidad / Relación (E/R). ....       | 23        |
| 3.1.2. Orientados a objetos. ....           | 24        |
| 3.2. Lógicos basados en registros. ....     | 24        |
| 3.2.1. Relacional. ....                     | 24        |
| 3.2.2. Red. ....                            | 24        |
| 3.2.3. Jerárquico. ....                     | 25        |
| 4. Lenguajes de datos. ....                 | 26        |
| 4.1. Utilidad. ....                         | 26        |
| 4.2. Dialectos. ....                        | 26        |
| 4.3. Partes. ....                           | 26        |
| 4.3.1. LDD. ....                            | 26        |
| 4.3.2. LMD. ....                            | 27        |

|         |   |    |
|---------|---|----|
| 5.      | Transacciones.....                                    | 27 |
| 5.1.    | Concepto.....   | 27 |
| 5.2.    | Características.....                                  | 27 |
| 5.3.    | Control de Concurrencia.....                          | 27 |
| 6.      | Usuarios.....   | 28 |
| 6.1.    | Administrador.....                                    | 28 |
| 6.2.    | Otros usuarios.....                                   | 28 |
| 7.      | Estructura general del sistema.....                   | 28 |
| 7.1.    | Organización / Interacción.....                       | 28 |
| 7.2.    | Componentes funcionales.....                          | 28 |
| 7.3.    | Estructuras de datos.....                             | 29 |
| TEMA 2. | MODELO ENTIDAD RELACIÓN (E/R).....                    | 31 |
| 1.      | Conceptos.....  | 31 |
| 1.1.    | Conjunto de entidades.....                            | 31 |
| 1.2.    | Conjunto de relaciones.....                           | 32 |
| 2.      | Cuestiones de diseño.....                             | 33 |
| 2.1.    | ¿Atributos o conjuntos de entidades?.....             | 33 |
| 2.2.    | ¿Conjunto de entidades o conjunto de relaciones?..... | 33 |
| 2.3.    | Conjunto de relaciones binarias frente a n-arias..... | 34 |
| 3.      | Ligaduras.....  | 35 |
| 3.1.    | Correspondencia de cardinalidad.....                  | 35 |
| 3.2.    | Dependencias de existencia.....                       | 36 |
| 4.      | Claves.....   | 37 |
| 4.1.    | Tipos de claves para conjuntos de entidades.....      | 37 |
| 4.2.    | Claves primarias para conjuntos de relaciones.....    | 37 |
| 5.      | El diagrama Entidad/Relación (E/R).....               | 37 |
| 5.1.    | Permite.....  | 37 |
| 5.2.    | Componentes.....                                      | 37 |
| 5.3.    | Detalles.....   | 38 |
| 6.      | Conjuntos de entidades débiles.....                   | 38 |
| 6.1.    | Definición.....                                       | 38 |
| 6.2.    | Características.....                                  | 39 |
| TEMA 3. | MODELOS E/R EXTENDIDO Y RELACIONAL.....               | 41 |
| 1.      | Modelo Entidad/Relación Extendido.....                | 41 |
| 1.1.    | Especialización.....                                  | 41 |
| 1.2.    | Generalización.....                                   | 42 |
| 1.3.    | Herencia.....   | 43 |
| 1.4.    | Ligaduras de diseño en especialización.....           | 44 |
| 1.5.    | Diagrama Entidad/Relación Extendido.....              | 45 |
| 1.6.    | Agregación.....                                       | 48 |
| 2.      | Reducción de un esquema E/R a relacional.....         | 49 |
| 2.1.    | Reducción de un conjunto de entidades fuerte.....     | 49 |
| 2.2.    | Reducción de un conjunto de entidades débiles.....    | 49 |
| 2.3.    | Reducción de un conjunto de relaciones.....           | 49 |
| 2.3.1.  | Caso general.....                                     | 49 |
| 2.3.2.  | Redundancia de tablas.....                            | 49 |



|  |    |
|--|----|
| 2.3.3. Combinación de tablas. ....                                 | 49 |
| 2.4. Reducción de los atributos multivalorados. ....               | 50 |
| 2.5. Reducción de una generalización. ....                         | 50 |
| 2.6. Reducción de una agregación. ....                             | 50 |
| 3. Consideraciones adicionales. ....                               | 51 |
| 3.1. Restricciones inherentes al modelo relacional. ....           | 51 |
| 3.2. Restricciones semánticas. ....                                | 51 |
| 3.3. Opciones de borrado y modificación en las claves ajenas. .... | 52 |
| 3.4. Otros mecanismos para representar las restricciones. ....     | 52 |
| 3.5. Algunas consideraciones sobre la notación utilizada. ....     | 52 |
| 4. El problema del estudiante. ....                                | 54 |
| 5. Problemas del Modelo Entidad/Relación. ....                     | 55 |
| 6. Soluciones propuestas a los ejercicios de E/R. ....             | 60 |
| TEMA 4. NORMALIZACIÓN .....  | 73 |
| 1. Terminología modelo relacional. ....                            | 73 |
| 2. Tablas planas. ....   | 73 |
| 2.1. Diseño. ....  | 73 |
| 2.2. Problemas. ....   | 74 |
| 3. Formas normales. ....   | 74 |
| 3.1. Primera forma normal. ....                                    | 74 |
| 3.2. Segunda forma normal. ....                                    | 75 |
| 3.3. Tercera forma normal. ....                                    | 75 |
| 3.4. Cuarta forma normal. ....                                     | 76 |
| 3.5. Quinta forma normal. ....                                     | 76 |
| 3.6. Forma normal de Boyce-Codd. ....                              | 76 |
| 4. Ventajas e inconvenientes. ....                                 | 77 |
| 5. Desnormalización. ....  | 77 |
| 5.1. Necesidad. ....   | 77 |
| 5.2. Reglas de la desnormalización. ....                           | 77 |
| 5.3. Técnicas. ....  | 78 |
| 5.3.1. Datos redundantes. ....                                     | 78 |
| 5.3.2. Columnas verticales. ....                                   | 79 |
| 5.3.3. Particionamiento horizontal. ....                           | 79 |
| 5.3.4. Particionamiento vertical. ....                             | 79 |
| TEMA 5. SQL .....  | 81 |
| 1. Introducción. ....  | 81 |
| 1.1. Estándar. ....  | 81 |
| 1.2. Evolución. ....   | 81 |
| 1.3. Componentes. ....   | 81 |
| 1.4. Herramientas. ....  | 82 |
| 2. SELECT. ....  | 82 |
| 2.1. Utilización básica. ....                                      | 82 |
| 2.2. Información de tablas, columnas y otros. ....                 | 83 |
| 2.3. <i>ORDER BY</i> . ....  | 83 |
| 3. WHERE. ....   | 84 |
| 3.1. Qué hace. ....  | 84 |

|        |   |     |
|--------|---|-----|
| 3.2.   | Condiciones de igualdad.....                                | 84  |
| 3.3.   | Condiciones de desigualdad. ....                            | 85  |
| 3.4.   | Lógica booleana.....  | 85  |
| 3.5.   | Between. ....   | 85  |
| 3.6.   | In.....   | 86  |
| 3.7.   | <i>Like</i> (o uso de comodines).....                       | 86  |
| 3.8.   | Valores Null.....   | 87  |
| 4.     | Columnas y funciones. ....                                  | 87  |
| 4.1.   | Manipulación de columnas.....                               | 87  |
| 4.2.   | Convert ( )......   | 87  |
| 4.3.   | Funciones.....  | 88  |
| 5.     | Agregados. ....   | 89  |
| 5.1.   | Agregados.....  | 89  |
| 5.2.   | Uso de los agregados.....                                   | 89  |
| 5.3.   | Con WHERE. ....   | 89  |
| 5.4.   | Con expresiones.....  | 90  |
| 5.5.   | Is Null ( , ). ....   | 90  |
| 6.     | Totales y subtotales. ....                                  | 90  |
| 6.1.   | Group By. ....  | 90  |
| 6.2.   | Cube y Rollup.....  | 91  |
| 6.3.   | Compute. ....   | 91  |
| 6.4.   | Having. ....  | 92  |
| 7.     | Un poco de todo. ....                                       | 92  |
| 7.1.   | Tablas de unión.....  | 92  |
| 7.2.   | Subconsultas. ....  | 93  |
| 7.3.   | Creación de tablas.....                                     | 94  |
| 7.4.   | Insertando y modificando datos: INSERT, UPDATE, DELETE..... | 95  |
| 8.     | Procedimientos Almacenados. ....                            | 97  |
| 8.1.   | Concepto.....   | 97  |
| 8.2.   | Creación.....   | 98  |
| 8.3.   | Información. ....   | 98  |
| 8.3.1. | Sobre un procedimiento creado.....                          | 98  |
| 8.6.   | Ejecución. ....   | 99  |
| 9.     | Disparadores (triggers).....                                | 101 |
| 9.1.   | Concepto.....   | 101 |
| 9.2.   | Información. ....   | 102 |
| 9.3.   | Funcionamiento de los Triggers. ....                        | 102 |
| 9.4.   | Aplicaciones. ....  | 103 |
| 10.    | Transact SQL - Ejemplos ....                                | 107 |
| 10.1.  | Tablas y campos ....  | 108 |
| 10.2.  | Ejemplos con SELECT.....                                    | 110 |
| 10.3.  | Ejemplos con WHERE ....                                     | 119 |
| 10.4.  | Columnas.....   | 122 |
| 10.5.  | Agregados.....  | 123 |
| 10.6.  | Totales y subtotales ....                                   | 125 |
| 10.7.  | Un poco de todo.....  | 130 |
| 10.8.  | Procedimientos Almacenados y Disparadores ....              | 133 |

|  |     |
|--|-----|
| 11. Transact SQL. Ejercicios propuestos. ....                  | 136 |
| 12. Soluciones a los Ejercicios propuestos. ....               | 138 |
| TEMA 6. ÍNDICES Y FUNCIONES DE ASOCIACIÓN.....                 | 151 |
| 1. Introducción. ....  | 151 |
| 1.1. Situación. ....   | 151 |
| 1.2. Métodos de acceso básicos. ....                           | 152 |
| 2. Índices ordenados. ....                                     | 152 |
| 2.1. Índice primario. ....                                     | 153 |
| 2.1.1. Índices densos y dispersos. ....                        | 154 |
| 2.1.2. Índices multinivel o jerarquizados. ....                | 155 |
| 2.1.3. Actualización de índices primarios. ....                | 156 |
| 2.2. Índices secundarios (o no agrupados). ....                | 156 |
| 2.3. Árboles equilibrados. ....                                | 157 |
| 2.3.1. Estructura de un nodo. ....                             | 157 |
| 2.3.2. Propiedades de un árbol B+. ....                        | 158 |
| 2.3.3. Consultas con árboles B+. ....                          | 158 |
| 2.3.4. Actualizaciones en árboles B+. ....                     | 159 |
| 3. Índices asociativos (Hashing). ....                         | 160 |
| 3.1. Asociación estática. ....                                 | 160 |
| 3.1.1. Concepto de cajón. ....                                 | 160 |
| 3.1.2. Función de asociación ( <i>hash</i> ). ....             | 161 |
| 3.1.3. Desbordamiento de cajones. ....                         | 162 |
| 3.2. Asociación dinámica. ....                                 | 164 |
| 3.2.1. Asociación extensible. ....                             | 164 |
| 4. Recomendaciones de uso. ....                                | 166 |
| 4.1. Algoritmos para insertar y borrar claves en árboles. .... | 166 |
| 4.1.1. Insertar. ....  | 166 |
| 4.1.2. Borrar. ....  | 167 |
| TEMA 7. TRANSACCIONES.....                                     | 169 |
| 1. Introducción. ....  | 169 |
| 1.1. Recuperación. ....  | 169 |
| 1.2. Observaciones. ....                                       | 169 |
| 1.2.1. Diferencia entre consistente y correcto. ....           | 170 |
| 2. Transacciones. ....   | 170 |
| 2.1. Concepto. ....  | 170 |
| 2.2. Ejemplo. ....   | 170 |
| 2.3. Información sobre GOTO y RETURN. ....                     | 170 |
| 2.4. Observaciones. ....                                       | 171 |
| 3. Recuperación de transacciones. ....                         | 171 |
| 3.1. Estructura. ....  | 171 |
| 3.2. Cuando se establece un punto de confirmación. ....        | 172 |
| 3.3. Unidad de recuperación. ....                              | 172 |
| 3.4. Propiedades PACA (ACID). ....                             | 172 |
| 4. Recuperación del sistema. ....                              | 173 |
| 4.1. Fallos durante la ejecución. ....                         | 173 |
| 4.2. Punto de verificación. ....                               | 173 |

|  |            |
|--|------------|
| 4.3. Procedimiento en el reinicio.....                   | 174        |
| 4.3.1. Identificación.....                               | 174        |
| 4.3.2. Procesado.....                                    | 174        |
| <b>TEMA 8. CONCURRENCIA .....</b>                        | <b>175</b> |
| 1. Introducción.....                                     | 175        |
| 2. Problemas de la concurrencia.....                     | 175        |
| 2.1. La actualización perdida.....                       | 176        |
| 2.2. La dependencia no confirmada.....                   | 176        |
| 2.3. El análisis inconsistente.....                      | 177        |
| 3. Bloqueos.....   | 178        |
| 3.1. Funcionamiento de los bloqueos.....                 | 178        |
| 3.1.1. Tipos de bloqueos.....                            | 178        |
| 3.1.2. Matriz de compatibilidad de tipos de bloqueo..... | 178        |
| 3.2. Protocolo de acceso a datos.....                    | 179        |
| 4. Nuevamente los tres problemas.....                    | 179        |
| 4.1. La actualización perdida.....                       | 179        |
| 4.2. La dependencia no confirmada.....                   | 180        |
| 4.3. El análisis inconsistente.....                      | 182        |
| 4.4. Bloqueo mortal.....                                 | 182        |
| 4.5. Seriabilidad.....                                   | 183        |
| 4.5.1. Concepto.....                                     | 183        |
| 4.5.2. Plan.....   | 184        |
| 4.5.3. Teorema del bloqueo de dos fases.....             | 184        |
| <b>PRÁCTICAS. MICROSOFT SQL SERVER 7.0 (MSS7) .....</b>  | <b>187</b> |
| <b>PRÁCTICA 1. VISIÓN GENERAL.....</b>                   | <b>189</b> |
| 1. Objetivos.....  | 189        |
| 2. Detalles.....   | 190        |
| 2.1. Cuestiones generales.....                           | 190        |
| 2.2. Características de la implementación.....           | 190        |
| 2.3. Herramientas gráficas.....                          | 191        |
| 2.3.1. Administrador corporativo.....                    | 191        |
| 2.3.2. Analizador de consultas.....                      | 191        |
| 2.3.3. Herramienta de red del cliente.....               | 191        |
| 2.3.4. Herramienta de red del servidor.....              | 191        |
| 2.3.5. Monitor de sistema de Windows 2000 (y de XP)..... | 191        |
| 2.3.6. Analizador de SQL Server.....                     | 192        |
| 2.3.7. Administrador de servicios.....                   | 192        |
| <b>PRÁCTICA 2. VISIÓN GENERAL (CONTINUACIÓN).....</b>    | <b>193</b> |
| 1. Objetivos.....  | 193        |
| 2. Detalles.....   | 194        |
| 2.1. Información del sistema.....                        | 194        |
| 2.2. Almacenamiento, datos y objetos.....                | 194        |
| 2.3. Diseñadores, diagramas, índices.....                | 195        |

|  |     |
|--|-----|
| PRÁCTICA 3. DISEÑO Y CREACIÓN DE BASES DE DATOS .....                | 197 |
| 1. Objetivos. ....   | 197 |
| 2. Detalles.....   | 198 |
| 2.1. Claves, Transacciones y Bloqueos. ....                          | 198 |
| 2.2. Diseño de Bases de Datos.....                                   | 198 |
| 2.3. Creación de Bases de Datos. ....                                | 199 |
| PRÁCTICA 4. BASE DE DATOS “LIGAINTERNA” .....                        | 201 |
| 1. Objetivos. ....   | 201 |
| 2. Detalles.....   | 201 |
| 2.1. Situación y aclaraciones. ....                                  | 201 |
| 2.1.1. Situación del problema. ....                                  | 201 |
| 2.1.2. Aclaraciones. ....  | 202 |
| 2.2. Enunciado del problema. ....                                    | 202 |
| 3. Tareas a realizar. ....   | 203 |
| PRÁCTICA 5. BASE DE DATOS “EXOTICFOOD” (I).....                      | 205 |
| 1. Objetivos. ....   | 205 |
| 2. Detalles.....   | 205 |
| 2.1. Descripción general. ....                                       | 205 |
| 2.2. Descripción de los datos. ....                                  | 206 |
| 2.2.1. Relacionados con los productos. ....                          | 206 |
| 2.2.2. Relacionados con los empleados.....                           | 208 |
| 2.2.3. Relacionados con los pedidos.....                             | 210 |
| 3. Se pide.....  | 212 |
| PRÁCTICA 6. BASE DE DATOS “EXOTICFOOD” (II) .....                    | 215 |
| 1. Objetivos. ....   | 215 |
| 2. Detalles.....   | 215 |
| 2.1. Opciones en el diseño de la Base de Datos. ....                 | 215 |
| 2.1.1. Entidades: CATEGORÍAS, PRODUCTOS y relación<br>CAT-PRO.....   | 216 |
| 2.1.2. PROVEEDORES, PRODUCTOS; PROVEE-PRODUC. ....                   | 216 |
| 2.1.3. PRODUCTOS; PEDIDOS, DETALLES PEDIDOS. ....                    | 216 |
| 2.1.4. EMPLEADOS. ....   | 216 |
| 2.1.5. CLIENTES, EMPLEADOS, TRANSPORTISTA;<br>PEDIDOS. ....          | 217 |
| 2.1.6. EMPLEADOS, TERRITORIOS, REGIONES; TER-EMP. ....               | 217 |
| 2.2. Tipos de datos utilizados. ....                                 | 217 |
| 2.2.1. Relacionados con los productos. ....                          | 217 |
| 2.2.2. Relacionados con los empleados.....                           | 218 |
| 2.2.3. Relacionados con los pedidos.....                             | 220 |
| 3. Relaciones "Exotic Food".....                                     | 222 |
| 4. Tipos de datos.....   | 223 |
| PRÁCTICA 7. “EXOTICFOOD” (III). RESTRICCIONES, VISTAS E ÍNDICES..... | 225 |
| 1. Objetivos. ....   | 225 |
| 2. Detalles.....   | 225 |
| 2.1. Restricciones de dominio. ....                                  | 225 |

|   |     |
|---|-----|
| 2.1.1. Restricciones CHECK. ....  | 225 |
| 2.1.2. Creación de restricciones CHECK. ....  | 226 |
| 2.2. Creación de vistas. ....   | 227 |
| 2.2.1. Información procedente de los Libros en Pantalla. ....                           | 227 |
| 2.2.2. Creación de vistas. ....   | 228 |
| 2.3. Índices. ....  | 229 |
| 2.3.1. Información sobre índices. ....  | 229 |
| 2.3.2. Crear índices en ExoticFood. ....  | 229 |
| 2.3.3. Responda a las siguientes cuestiones. ....                                       | 230 |
| PRÁCTICA 8. “EXOTICFOOD” (IV). SQL Y EL ANALIZADOR DE CONSULTAS .....                   | 231 |
| 1. Objetivos. ....  | 231 |
| 2. Detalles. ....   | 231 |
| 2.1. Creación de tablas. ....   | 231 |
| 2.2. Modificación de tablas. ....   | 234 |
| 2.3. Creación de vistas. ....   | 234 |
| 2.4. Insertar datos. ....   | 235 |
| 2.5. Consultas. ....  | 235 |
| PRÁCTICA 9. CREACIÓN DE APLICACIONES DE BASES DE DATOS CON<br>BORLAND BUILDER C++ ..... | 237 |
| 1. Objetivos. ....  | 237 |
| 2. Detalles. ....   | 237 |
| 2.1. Importar una base de datos externa. ....   | 237 |
| 2.2. Crear un nuevo origen de datos. ....   | 238 |
| 2.3. Crear una nueva aplicación. ....   | 239 |
| 2.4. Añadir componentes de bases de datos. ....   | 239 |
| 2.5. Componentes de conexión a la base de datos. ....                                   | 239 |
| 2.6. Añadiendo acciones estándar. ....  | 240 |
| 2.7. Añadiendo un menú. ....  | 240 |
| 2.8. Añadiendo una barra de herramientas. ....  | 241 |
| 2.9. Añadiendo controles para la tabla EMPLOYEE. ....                                   | 242 |
| 2.10. Añadiendo controles para la tabla ITEMS. ....                                     | 242 |
| TIPOS DE DATOS. M. SQL SERVER 7.0 .....   | 243 |
| 1. Especificar un tipo de datos para una columna. ....                                  | 251 |
| 2. Exigir la integridad de los datos. ....  | 251 |
| 2.1. Datos binarios. ....   | 251 |
| 2.2. Datos de carácter. ....  | 252 |
| 2.3. Datos Unicode. ....  | 252 |
| 2.4. Datos de fecha y hora. ....  | 253 |
| 2.5. Datos numéricos. ....  | 253 |
| 2.5.1. Datos enteros. ....  | 253 |
| 2.5.2. Datos decimales. ....  | 253 |
| 2.5.3. Datos numéricos aproximados. ....  | 254 |
| 2.6. Datos de moneda. ....  | 254 |
| 2.7. Datos especiales. ....   | 254 |
| 3. Crear tipos de datos definidos por el usuario. ....                                  | 255 |

|                                     |            |
|-------------------------------------|------------|
| <b>BIBLIOGRAFÍA .....</b>           | <b>257</b> |
| 1. Bibliografía básica. ....        | 259        |
| 2. Bibliografía complementaria..... | 261        |
| 3. Bibliografía auxiliar. ....      | 262        |





TEORÍA



# TEMA 1. INTRODUCCIÓN

## 1. INTERÉS DE LAS BASES DE DATOS.

### 1.1. SIN BASES DE DATOS.

Situación: una entidad bancaria necesita una aplicación para poder almacenar información sobre los clientes, los empleados, los distintos productos financieros que se ofrecen, las sucursales pertenecientes a la entidad, las cuentas abiertas, etc. Con esta misma aplicación se deben poder realizar operaciones de consulta, actualización, etc., de los datos anteriormente mencionados. Una solución propuesta por un técnico que desconozca los principios de las bases de datos podría ser la siguiente:

#### *Inicialmente*

A partir de las especificaciones del problema anterior, se identifican los datos a almacenar:

- Los clientes.
- Los empleados.
- Las cuentas.
- Los distintos productos financieros.

| Clientes |           |           |             |              |
|----------|-----------|-----------|-------------|--------------|
| Nombre   | Apellidos | Dirección | Teléfono    | Nº de cuenta |
| Pepe;    | López;    | Díaz;     | 987 271235  | 10000001;    |
| Juan;    | García;   | Alvarez;  | 987 201015; | 10000002;    |
| • • •;   |           |           | 555 333;    | 10000003;    |

| <b>Productos</b>             |            |               |                    |                 |                      |
|------------------------------|------------|---------------|--------------------|-----------------|----------------------|
| <i>Tipo producto</i>         | <i>TIN</i> | <i>Apunte</i> | <i>Descubierto</i> | <i>Apertura</i> | <i>Mantenimiento</i> |
| Cuenta ahorro;               | 0,25;      | 0,30;         | 12 %;              | 6 €;            | 4 €;                 |
| Libretín;                    | 0,05;      | 0,10;         | 9 %;               |                 | 9 €;                 |
| Cuenta<br>superahorro;       |            | 0,60;         |                    |                 | 2 €;                 |
| Cuenta para ricos<br>(CPR);  | 2;         |               | 22 %;              | 15 €;           | 14 €;                |
| Cuenta para pobres<br>(CPP); | 0,05;      | 0,09;         | 6 %;               |                 |                      |
| • • •                        |            |               |                    |                 |                      |

| <b>Empleados</b> |                  |                |              |                  |               |
|------------------|------------------|----------------|--------------|------------------|---------------|
| <i>Nombre</i>    | <i>Apellidos</i> | <i>Direcc.</i> | <i>Tfno.</i> | <i>Categoría</i> | <i>Nómina</i> |
| Jose;            |                  |                |              | Cajero;          | 725;          |
| • • •            |                  |                |              | Ingeniero;       | 1100;         |

| <b>Cuenta cliente</b> |                 |                  |              |
|-----------------------|-----------------|------------------|--------------|
| <i>Num. Cuenta:</i>   | <i>Debe (€)</i> | <i>Haber (€)</i> | <i>Saldo</i> |
| 10000001              | -40;            |                  | 450;         |
| 10000001              |                 | 120;             | 570;         |

En un primer momento se crean distintos ficheros de consulta en los que se almacena la información, como son:

- Fichero de consulta de clientes: para obtener datos como el nombre, los apellidos, el teléfono, etc.
- Fichero de consulta de los distintos productos financieros, donde se almacenan datos como el interés, la comisión, etc.
- Fichero de consulta de información de los empleados donde se guarda información como el nombre, los apellidos, la dirección, la nómina, etc.
- Fichero con la información de las cuentas de los clientes. Se almacenan datos como la fecha de apertura, el saldo actual, los movimientos realizados, etc.

A partir de los ficheros anteriores se crean cuatro aplicaciones distintas para gestionar la información:

- Creación de las nuevas cuentas.

- Actualización del estado de las cuentas.
- Gestión de clientes.
- Creación de informes.

### ***Ampliaciones posteriores***

Un año después de la implantación de la aplicación, aparecen las siguientes modificaciones respecto a las especificaciones que se proporcionaron inicialmente:

- Ha aumentado el número de oficinas.
- Se han creado nuevas aplicaciones financieras.
- Ha aumentado el número de clientes.

## **1.2. INCONVENIENTES.**

Tras un cierto tiempo de prueba de la aplicación, se han detectado los siguientes problemas:

- a) Se tienen datos repetidos y no son iguales; problema de redundancia e inconsistencia.
- b) Acceso complicado a datos internos (necesidades no previstas).
- c) Hay archivos con distintos formatos; problema del aislamiento. Es difícil agrupar datos de archivos diferentes y el acceso a ellos también es más complejo.
- d) Integridad de nuevas ligaduras (las normas y los permisos pueden variar).
- e) Si el sistema se cae, existen errores en las cuentas; problema de atomicidad. Es difícil tener transacciones con sistemas de archivos.
- f) El cajero conoce tu nómina; problema de seguridad. No existen vistas ni restricción de acceso a los datos.
- g) El misterio de los Pérez. A partir de una sola cuenta con 100 €, en un momento dado, los tres sacan dinero al mismo tiempo –cada uno 10 €–, pero en la cuenta quedan 90 €. El problema está en que los tres acceden a la misma información inicial (no hay bloqueos).

## **1.3. SOLUCIÓN.**

En los sistemas de archivos aparecen los problemas anteriormente expuestos. Al crecer, se hacen más complejos debido a que son rígidos. La solución a todos estos problemas es tener un sistema gestor de bases de datos (S.G.B.D.).

# **2. VISIÓN DE LOS DATOS.**

## **2.1. SISTEMA GESTOR DE BASES DE DATOS.**

Consiste en un conjunto de programas que permiten actuar sobre una base de datos para modificar, insertar y buscar datos interrelacionados.

Presenta una serie de características:

- Abstracciones en el diseño de los datos.
- Datos homogéneos: para que no queden aislados.
- Mecanismo de control<sup>1</sup>:
  - Bloqueos.
  - Transacciones.
  - Seguridad: tanto de acceso como para facilitar la realización de copias de seguridad.

## 2.2. ABSTRACCIÓN.

Es un mecanismo que permite esconder la complejidad; para ello se crean tres niveles o esquemas.

- 1) Nivel físico: contiene las estructuras de datos, cómo están almacenados físicamente.
- 2) Nivel lógico: datos que hay y relaciones entre ellos.
- 3) Nivel de vistas: qué datos se pueden ver según quien seamos.

El nivel físico lo gestionan los diseñadores de sistemas gestores de bases de datos, eligiendo la estructura física. El diseñador de la base de datos elige los datos y sus relaciones. En el nivel de vistas, dado un conjunto de datos se determina quién los puede ver y quién no. De esta forma se pueden tener, en el mismo nivel lógico, varias vistas.

## 2.3. EJEMPLAR Y ESQUEMA.

Un **ejemplar** es un conjunto de datos que hay en un momento dado en nuestro sistema (los datos varían).

Un **esquema** es la estructura de los datos. Hay tres tipos de esquemas:

- Físico: indica cómo se tienen los datos organizados.
- Lógico: indica la estructura de la base de datos.
- Vistas que se tienen.

Cada ejemplar tiene un esquema. Un esquema, una vez realizado y aprobado, no se cambia nunca ya que es muy costoso. Debe diseñarse correctamente desde el principio.

## 2.4. INDEPENDENCIA DE DATOS.

Capacidad de modificar esquemas en un nivel sin que esto afecte a un nivel más alto (vistas o aplicación).

El nivel lógico nunca lo debe manejar el usuario; para eso está el nivel de vistas.

Existen dos tipos de independencia:

---

<sup>1</sup> Este no siempre aparece.

- Física: capacidad de modificar el nivel físico sin afectar al de vistas.
- Lógica: capacidad de modificar el esquema lógico sin afectar al de vistas (o aplicaciones, interfaz, etc.).

### 3. MODELOS DE DATOS.

Son herramientas conceptuales que permiten definir los datos, las relaciones entre los datos, las ligaduras que existen en el problema y la semántica.

Hay dos tipos de modelos:

- Lógicos: orientados a objetos o a registros.
- Físicos.

#### 3.1. LÓGICOS BASADOS EN OBJETOS.

Se utilizan para describir los datos en los niveles lógico y de vistas. Se caracterizan por tener una estructura flexible y porque permiten explicitar las ligaduras. Existen tres tipos:

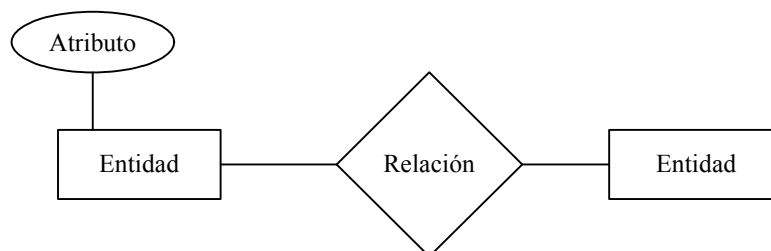
- E/R (Entidad / Relación).
- Orientados a objetos.
- De datos: semántico y funcional.

##### 3.1.1. Entidad / Relación (E/R).

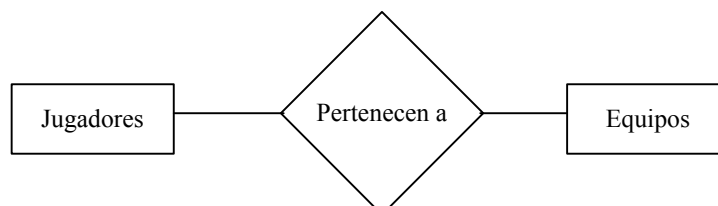
Modela los datos mediante entidades y los vínculos entre datos con interrelaciones.

Def.- **Entidad**: objeto con un conjunto de valores que tiene sentido por sí mismo (*con vida propia*).

Los diagramas E/R se utilizan para representar el modelo E/R gráficamente.



Ejemplo:



Las relaciones también pueden tener atributos.

### 3.1.2. Orientados a objetos.

Están basados en una colección de objetos. Los objetos (o datos) tienen diversas propiedades y métodos (accesibles sólo desde dentro del mismo). Los objetos similares se agrupan en clases. Un dato concreto es una instancia de un objeto.

Además:

- Disponen de encapsulación;
- La forma de comunicarse con un objeto es mediante la llamada a sus métodos externos.

Todo esto genera un nivel de abstracción que facilita el modelado de problemas.

Están recomendados para tratar datos de difícil clasificación por propiedades: elementos multimedia, imágenes, etc. A pesar de todo, el que más se utiliza es el modelo relacional.

## 3.2. LÓGICOS BASADOS EN REGISTROS.

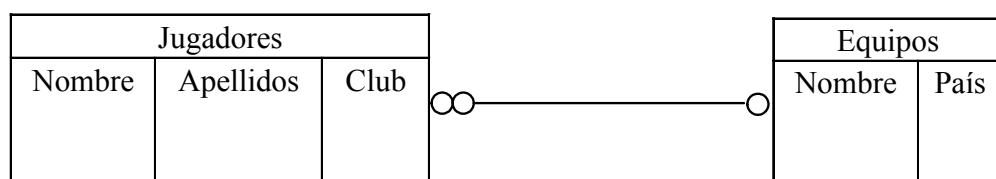
Se utilizan para describir datos a nivel de vistas y a nivel lógico. Se denominan así porque la base de datos está estructurada en registros de estructura fija. A su vez, los registros están formados por campos que tienen un tipo de datos y una longitud constante.

Los tipos son:

- Relacional.
- Red.
- Jerárquico.

### 3.2.1. Relacional.

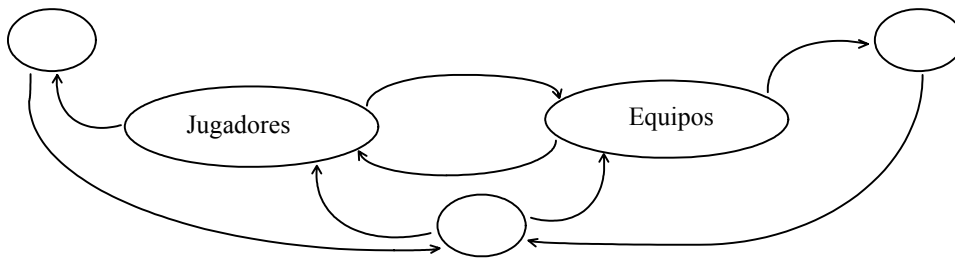
La base de datos está formada por una colección de tablas y relaciones entre las tablas. Cada tabla tiene un número de columnas constante y de longitud constante. Cada registro es un fila o tupla, y está formado por campos.



### 3.2.2. Red.

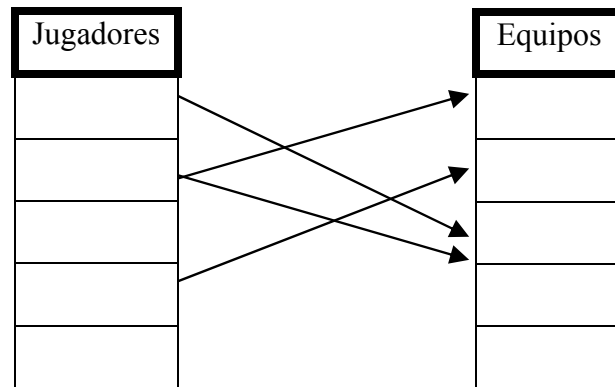
En este modelo las entidades se representan como nodos de un grafo y las asociaciones entre entidades se reflejan mediante arcos.





Físicamente los datos se disponen como una colección de registros, y los relacionamos con punteros entre registros.

Por ejemplo, tenemos una colección de registros llamada “Jugadores” y otra llamada “Equipos”. Cada jugador tendrá un puntero al equipo que pertenece:

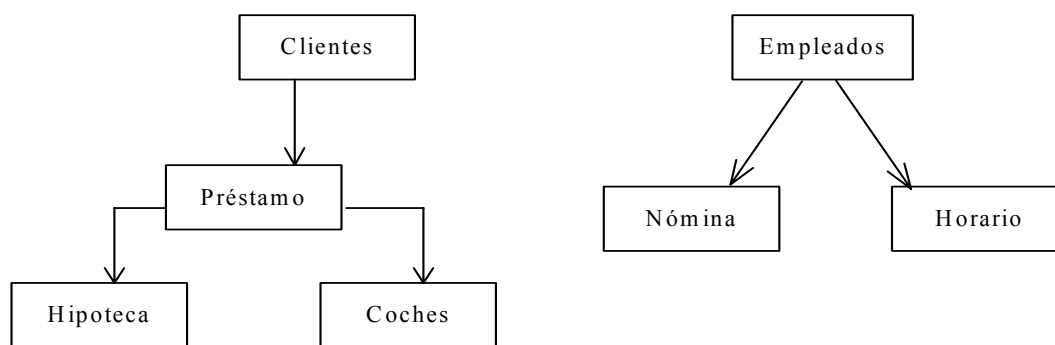


Las características más importantes son:

- Cada registro tiene un puntero a cada uno de los registros a los que está asociado, pudiendo llegar y salir varios punteros.
- Un nodo puede tener varios padres.

### 3.2.3. Jerárquico.

Es similar al modelo en red. En este caso los datos y las relaciones que hay se expresan, respectivamente, mediante registros y punteros. Los registros están organizados como colecciones de árboles. Es una particularización del modelo en red: cada nodo no puede tener más de un padre.



El modelo en red y el jerárquico son los más rápidos en cuestión de accesos; sin embargo el modelo relacional es el más flexible. En general el que más se utiliza es el modelo relacional.

## 4. LENGUAJES DE DATOS.

### 4.1. UTILIDAD.

Sirven para crear y modificar el esquema de la base de datos y para acceder a los datos (manipularlos); esto es crearlos, consultarlos y actualizarlos.

### 4.2. DIALECTOS.

En 1970 IBM creó el SEQUEL, y debido a su éxito se convierte, casi una década después, en un estándar: el SQL. Diez años más tarde aparecieron, principalmente, el “SQL2” y el “SQL ANSI 92”. Otros lenguajes son el QBE (aquí la consulta se realiza mediante ejemplos de lo que se quiere encontrar) y el DataLog. El SQL es el más utilizado en los modelos relacionales.

Los dialectos incorporan todas las características del modelo básico y añaden algunas nuevas. Algunos dialectos del SQL son el “Transact SQL” (de MS SQL Server) y el PL/SQL (de Oracle).

### 4.3. PARTES.

- Lenguaje de definición de datos (LDD): para crear y modificar datos.
- Lenguaje de manipulación de datos (LMD): para acceder a ellos.

#### 4.3.1. LDD.

Sirve para crear los esquemas (el lógico y el de vistas básicamente). Las instrucciones principales son:

- create database;
- create tabla;
- create view;
- constraint, primary key, etc.

Cuando se compilan las instrucciones del LDD se produce un conjunto de tablas que se almacenan en un archivo al que se le denomina “diccionario de datos”; se dice

que contiene metadatos (datos de datos), es decir, información sobre la estructura. Se consulta o lee antes de modificar datos.

#### 4.3.2. LMD.

Sirve para manipular los datos, es decir, realiza cuatro operaciones (MIBoR):

- Modificación (update);
- Inserción (insert);
- Borrado (delete);
- Recuperación (select).

Hay dos tipos de LMD:

- *Procedimentales*: aquí es necesario que el usuario indique qué datos quiere y cómo recuperarlos.
- *No procedimentales*: el usuario especifica qué datos quiere recuperar (sin indicar cómo). Éstos son más fáciles de utilizar y de aprender, pero normalmente son menos eficientes.

Un subconjunto de LMD es el lenguaje de consulta, que se representa mediante la cláusula `Select`.

## 5. TRANSACCIONES.

### 5.1. CONCEPTO.

Son unidades lógicas de trabajo, formadas habitualmente por un conjunto de operaciones que han de ejecutarse como si de una única operación se tratara.

### 5.2. CARACTERÍSTICAS.

- *Atomicidad*: las transacciones se ejecutan como si fuesen una unidad.
- *Consistencia* (congruencia): tras la ejecución de la transacción, el estado de la base de datos ha de ser congruente con su estado anterior.
- *Independencia*: las transacciones son independientes unas de otras.
- *Durabilidad*: tras la ejecución, los cambios que se efectúen tienen que permanecer en el tiempo.

### 5.3. CONTROL DE CONCURRENCIA.

El sistema gestor de bases de datos es multiusuario, por lo que hay usuarios concurrentes. El control de la concurrencia se realiza mediante bloqueos y sirve para controlar la interacción entre transacciones concurrentes. Sólo se permite que una transacción acceda a un conjunto de datos si esos datos están bloqueados por la transacción.

Hay varios tipos de bloqueos:

- Compartido: si la transacción  $T_i$  tiene bloqueado de forma compartida el conjunto de datos  $Q$ , esto implica que  $T_i$  puede leer pero no escribir. También permite que otra transacción pueda leer de forma simultánea.
- Exclusivo: si la transacción  $T_i$  tiene un bloqueo exclusivo sobre el conjunto de datos  $Q$ , puede leer y escribir, pero ninguna otra transacción puede acceder a esos datos.
- Otros.

Las transacciones han de ser rápidas, sin estados intermedios y sin compartir datos para que el sistema sea eficiente.

## 6. USUARIOS.

### 6.1. ADMINISTRADOR.

Es el que se encarga de generar y modificar los esquemas (lógico y de vistas) de la base de datos, así como de las cuestiones de seguridad (control de acceso y recuperación/almacenamiento de datos). El almacenamiento y recuperación se realiza con una política de copias de respaldo (normalmente en cintas) e implementando sistemas de redundancia RAID (varios dispositivos físicos anticáida del sistema).

### 6.2. OTROS USUARIOS.

- *Programadores de aplicaciones*: crean aplicaciones que acceden a los datos de la base de datos mediante el LMD embebido en los lenguajes de programación.
- *Usuarios sofisticados*: utilizan consultas mediante LMD, utilizando en una interfaz de comandos instrucciones de SQL puro.
- *Programadores de aplicaciones especiales*: diseñan aplicaciones de CAD, sistemas expertos o bases de conocimiento.
- *Usuarios normales*: acceden a la base de datos mediante las aplicaciones que los programadores han diseñado para tal efecto.

## 7. ESTRUCTURA GENERAL DEL SISTEMA.

### 7.1. ORGANIZACIÓN / INTERACCIÓN.

El sistema gestor de bases de datos divide el trabajo en tareas, de las cuales unas las realizará el sistema operativo y otras, el gestor. Para la comunicación entre el gestor y el sistema operativo existen módulos.

### 7.2. COMPONENTES FUNCIONALES.

El sistema gestor de bases de datos está formado por:

- a) Módulo de procesamiento de consultas, en el que hay:
  - Compilador del LMD.
  - Precompilador del LMD incorporado.

- Intérprete del LDD.
- Motor de evaluación de consultas.
- b) Módulo de gestión de almacenamiento, donde se necesitan:
  - Gestores de autorizaciones y seguridad.
  - Gestores de transacciones.
  - Gestores de archivos.
  - Gestores de memoria intermedia.

### **7.3. ESTRUCTURAS DE DATOS.**

Las estructuras de datos que tiene que manejar el sistema gestor de bases de datos son:

- Archivos de datos: con la información de las tablas.
- Índices.
- Diccionario de datos: con información de la base de datos.
- Datos estadísticos utilizados por el optimizador de consultas.



# TEMA 2. MODELO

## ENTIDAD RELACIÓN (E/R)

### 1. CONCEPTOS.

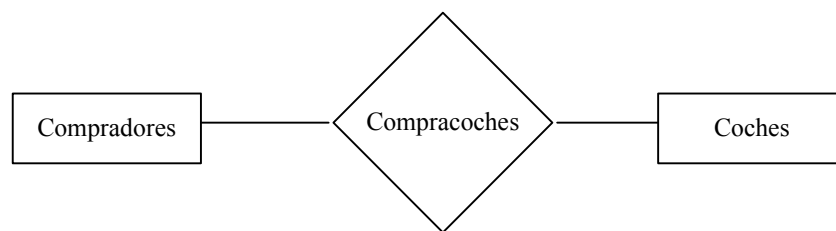
#### 1.1. CONJUNTO DE ENTIDADES.

- **Entidad:** Es un objeto diferenciable de otros, se puede decir que tiene 'vida propia'. Algunos ejemplos son: cliente, libro, asignatura. La propiedad fundamental de cualquier entidad es que se tiene que diferenciar unívocamente de todas aquellas entidades del mismo tipo.
- **Conjunto de entidades (c.d.e.):** Totalidad de las entidades que son del mismo tipo y que tienen los mismos atributos. Los conjuntos de entidades no tienen por qué ser necesariamente disjuntos.
  - **Extensión:** Entidad particular dentro de un conjunto de entidades.
  - **Atributos:** Valores asociados a cada una de las propiedades de una entidad. Toda entidad queda representada por medio de un conjunto de atributos. Los atributos tienen un *dominio* (conjunto de valores que puede tomar). El conjunto de pares atributo-valor sirven para describir a cada entidad.
    - ✓ Tipos de atributos:
      - *Simples:* Atributo formado por un único valor.
      - *Compuestos:* Atributo formado por varios valores que podemos descomponer.
      - *Univalorados:* Aceptan un único valor.

- *Multivalorados*: Aceptan varios valores.
- *Nulos*: Asociado a atributos que no son aplicados o no se conocen. Los valores NULL pueden dar problemas en operaciones de recuperación de datos.
- *Derivados*: Aquellos que se obtienen a partir de otros atributos. A veces aparecen por razones de eficiencia en las búsquedas. Tienen un tratamiento especial; se guardan en columnas calculadas, aunque lo habitual es que no se guarden.

## 1.2. CONJUNTO DE RELACIONES.

- ✓ Una **relación** consiste en la asociación entre dos o más entidades diferentes.

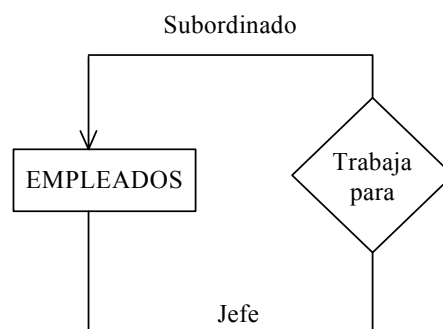


- ✓ Un **conjunto de relaciones** es la totalidad de las relaciones de un mismo tipo.
- ✓ Una **participación** es una asociación de un conjunto de entidades (c.d.e.)  $\{E_1, \dots, E_n\}$ , mediante un conjunto de relaciones (c.d.r.) R.  
Se dice que la entidad  $E_1$  *participa* en la relación R.
- ✓ Se denomina **ejemplar** a una relación concreta de un conjunto de relaciones.

El **papel** de la entidad:

- Si se tienen conjuntos de entidades distintas, el papel de la entidad suele ser obvio.
- Si hay conjuntos de entidades del mismo tipo el papel ya no es tan evidente.

Ejemplo:





Hay una jerarquía de empleados, diferenciándose en los subordinados que tengan. El conjunto de relaciones “Trabaja para” es reflexiva, con lo que el papel del conjunto de entidades “Empleados” no es obvio. En cambio, si se dice que un jefe puede tener varios empleados, su papel en la relación es más claro.

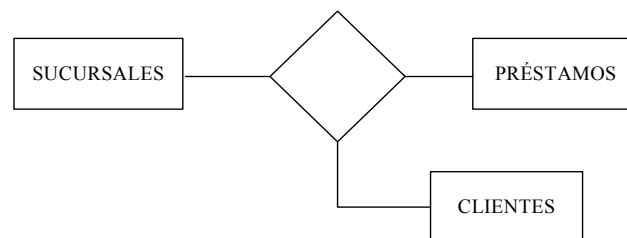
Características de los conjuntos de relaciones:

- Pueden tener *atributos descriptivos*.

Ejemplo: en la relación “compra coche” se puede tener el atributo “fecha de compra”.

- *Grado del conjunto de relaciones*: puede ser una relación binaria si participan 2 conjuntos de entidades; una relación ternaria será aquella en la que participan 3 conjuntos de entidades.

Ejemplo:



## 2. CUESTIONES DE DISEÑO.

### 2.1 ¿ATRIBUTOS O CONJUNTOS DE ENTIDADES?

Se tienen las siguientes dos situaciones:

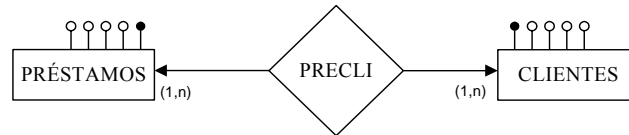
- (a) Dado un conjunto de empleados (entidades) se quiere guardar información sobre ellos (DNI, nombre, dirección y números de teléfono). Si interesa el atributo “teléfono” y se supone que un mismo teléfono no puede pertenecer a más de un empleado, en este caso se representaría el teléfono como un atributo multivalorado.
- (b) A partir de la misma situación del caso anterior se quiere guardar además el gasto mensual telefónico y la ubicación de cada teléfono. Suponiendo que un mismo teléfono puede pertenecer a varios empleados, el “teléfono” será descrito como una entidad.

Como norma general cuando un atributo crezca demasiado (por ejemplo, un atributo multivalorado con demasiadas opciones) se transformará en un conjunto de entidades.

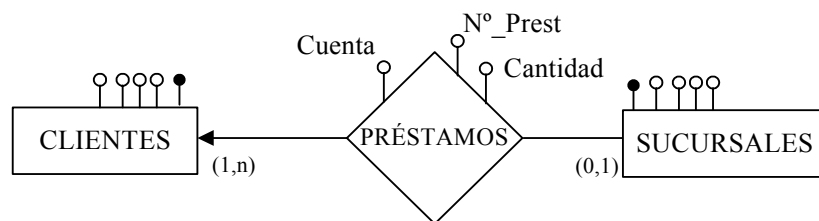
### 2.2 ¿CONJUNTO DE ENTIDADES O CONJUNTO DE RELACIONES?

Suponiendo el caso de los préstamos asignados a los clientes en una determinada sucursal:

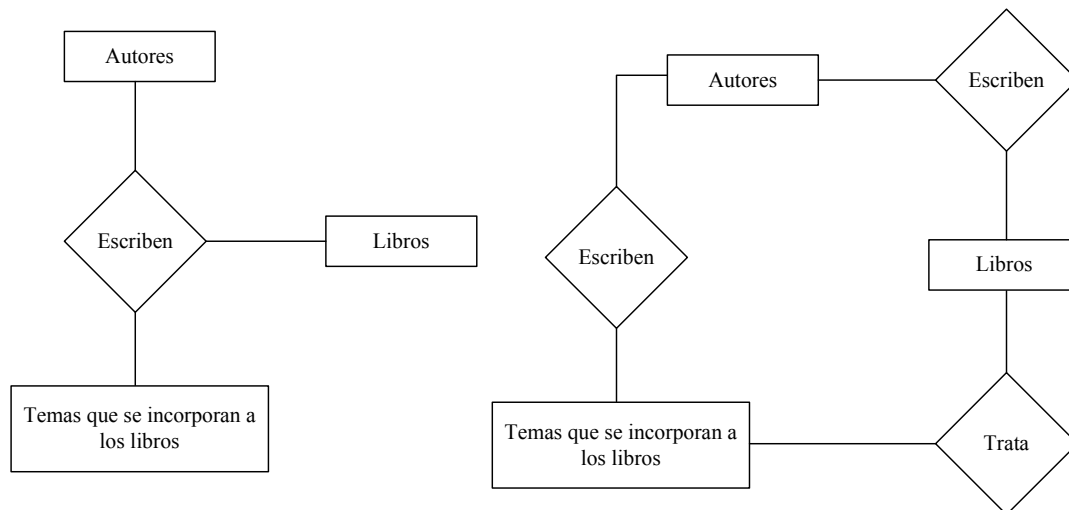
- (a) Se considera el préstamo como un conjunto de entidades. En este caso se refleja que un préstamo puede estar asociado a varios clientes (también a la inversa).



- (b) Se considera el préstamo como un conjunto de relaciones. En este caso un determinado préstamo sólo puede estar asociado a un cliente (y a la inversa).

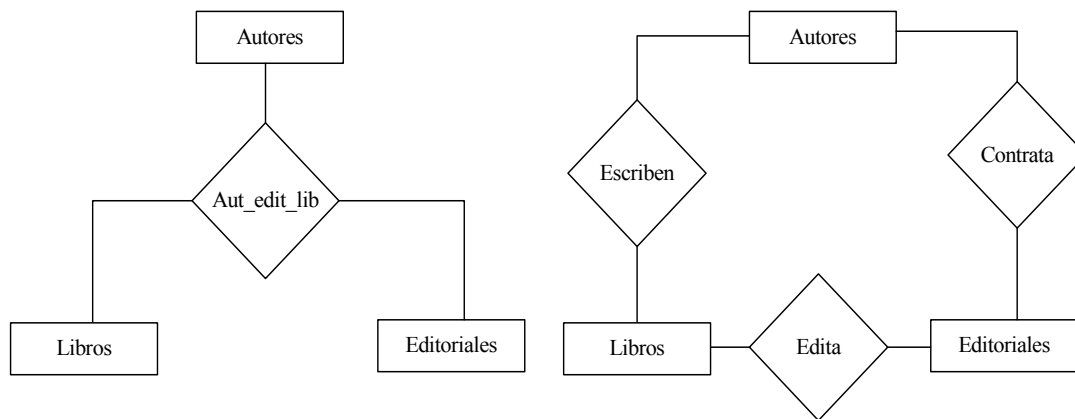


### 2.3. CONJUNTO DE RELACIONES BINARIAS FRENTE A N-ARIAS.



Se tiene el conjunto de entidades autores que escriben sobre libros y sobre temas que se incorporan a los libros, como se refleja en el diagrama de la izquierda. Al convertir la relación ternaria “Escriben” a binaria, como queda representado en el diagrama de la derecha, surge el siguiente problema: con la relación ternaria un libro puede estar escrito por un autor que además escribe sobre temas en un determinado

libro. Con la relación binaria no se sabe que el autor escribe un tema determinado de un libro, sólo que el autor escribe temas. Por ejemplo, un autor escribe un libro de biología y a su vez escribe un tema sobre ecología en dicho libro. Con la representación de la derecha, queda reflejado que el autor tiene un libro de biología y que, por otra parte, escribió temas sobre ecología, pero no indica que dichos temas están en el libro de biología. Por tanto, no se puede sustituir la relación ternaria por una binaria porque no recoge la semántica.



El conjunto de entidades “Autores” tiene relación con los conjuntos de entidades “Libros” y “Editoriales”, como refleja el diagrama de la izquierda. Al utilizar relaciones binarias queda el esquema de la derecha. Uno o varios autores se relacionan con un libro y con una editorial. Un libro lo edita una única editorial. Si el libro “x” está escrito por tres autores y editado por una editorial y, además, un autor está contratado por la editorial “y”, en este caso, los tres autores estarán contratados por dicha editorial “y”. Se mantiene la semántica.

Siempre que se pueda las relaciones utilizadas han de ser binarias. Esto es posible cuando dos relaciones binarias mantienen la misma semántica que la ternaria. Las relaciones ternarias (o en general n-arias) deben ser usadas cuando se desea evidenciar la participación de las tres entidades (o en general de las n entidades).

### 3. LIGADURAS.

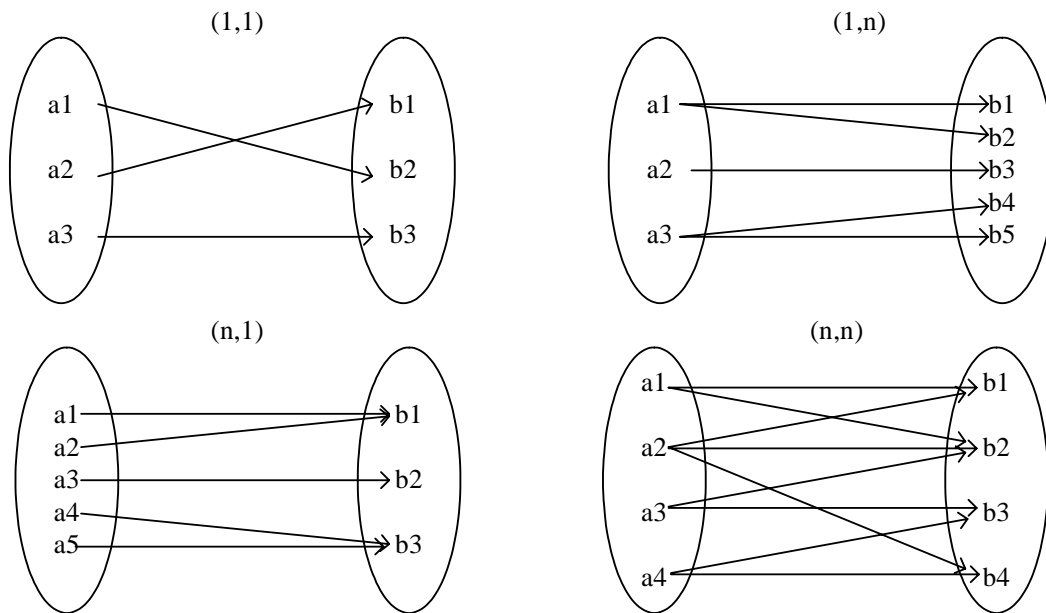
#### 3.1. CORRESPONDENCIA DE CARDINALIDAD.

Expresa el número de entidades que están asociadas a otra mediante un conjunto de relaciones.

Tipos de cardinalidades (para conjuntos de relaciones binarias):

- **Uno a uno** (1,1): una entidad de A se puede relacionar con una de B.
- **Uno a varios** (1,n): una entidad de A se puede relacionar con una o varias de B.
- **Varios a uno** (n,1): varias entidades de A se pueden relacionar con una de B.

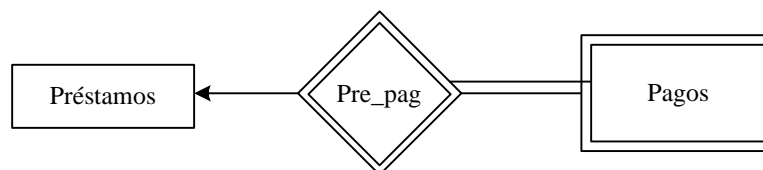
- **Varios a varios (n,n):** varias entidades de A se pueden relacionar con varias entidades de B.



### 3.2. DEPENDENCIAS DE EXISTENCIA.

La existencia de un conjunto de entidades  $x$  dependerá de la existencia de otro conjunto de entidades  $y$  cuando una entidad del primer conjunto de entidades no pueda existir si no existe una entidad del segundo conjunto de entidades. A  $x$  se le denomina conjunto de entidades **subordinadas** o **débiles**; a  $y$ , conjunto de entidades **dominantes**. Esta dependencia de existencia hace que si se borra el conjunto de entidades  $y$ , el conjunto de entidades  $x$  también sería borrado de forma automática.

Ejemplo de participación de un conjunto de entidades E en un conjunto de relaciones R:



“Pagos” es una entidad porque tiene una serie de atributos que le hacen tener vida propia por sí misma, pero no se puede pensar en un pago sin relacionarlo con un préstamo. Si los préstamos desaparecen, también los pagos. El doble rectángulo indica que es un conjunto de entidades débil.

Hay dos tipos de participación de un conjunto de entidades E en un conjunto de relaciones R:

- *Participación total:* Cada entidad de E participa al menos en una relación de R. Suele producir dependencia de existencia. Este tipo de participación se denota por una doble raya.

- *Participación parcial*: Las entidades de E no tienen porqué pertenecer a una relación de R. Este tipo de participación se denota mediante una línea simple.

## 4. CLAVES.

Una clave es un atributo que permite diferenciar una entidad (o relación) del resto dentro del mismo conjunto de entidades (o de relaciones).

### 4.1. TIPOS DE CLAVES PARA CONJUNTOS DE ENTIDADES.

- **Superclaves**: una clave será superclave si existen unos atributos que colectivamente identifican a una determinada identidad. Una superclave puede contener atributos que son innecesarios para identificar a una entidad. Este tipo de claves poseen propiedades de unicidad pero no de irreductibilidad.
- **Clave candidata**: es aquella superclave cuyos subconjuntos no son superclaves. Este tipo de claves cumplen propiedades de unicidad (no existen dos entidades con el mismo valor de clave candidata) y de irreductibilidad (ningún subconjunto de la clave candidata tiene propiedades de unicidad).
- **Clave primaria**: es la clave candidata que se elige para representar un determinado conjunto de entidades.

### 4.2. CLAVES PRIMARIAS PARA CONJUNTOS DE RELACIONES.

- Si una relación R no tiene atributos asociados, su clave primaria estará formada por la unión de todas las claves primarias de las entidades que participan en la relación R.
- Si una relación R tiene atributos asociados, la clave primaria de la relación estará formada por la unión de las claves primarias de las entidades que participan en la relación y los atributos de la relación R.

Dada una relación R en la que participan las entidades  $E_1 \dots E_n$ , la clave formada por la unión de las claves primarias de  $E_1 \dots E_n$  es una superclave.

## 5. EL DIAGRAMA ENTIDAD/RELACIÓN (E/R).

### 5.1. PERMITE.

El diagrama Entidad/Relación permite representar gráficamente la estructura lógica de la base de datos.

### 5.2. COMPONENTES.

Los componentes básicos de este diagrama son:

- Rectángulos.
  - Rectángulo simple: conjunto de entidades fuerte.
  - Rectángulo doble: conjunto de entidades débil.

- Elipses.
  - Elipse simple: atributo.
  - Elipse doble: atributo multivalorado.
  - Elipse discontinua: atributo derivado.
- Rombos.
  - Rombo simple: conjunto de relaciones.
  - Rombo doble: conjunto de relaciones en una entidad débil.
- Líneas.
  - Líneas simples: unen atributos con los conjuntos de entidades, atributos con los conjuntos de atributos o conjuntos de entidades con conjuntos de relaciones (relación de participación parcial).
  - Líneas dobles: indican una relación de participación total.

### 5.3. DETALLES.

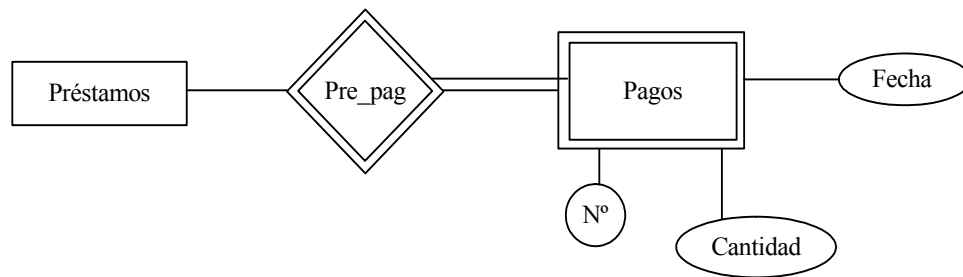
- La clave primaria de un conjunto de entidades se indica subrayando los atributos que la componen.
- Los atributos de las relaciones van unidos a los rombos que representan las relaciones mediante una línea simple.
- Cuando sea necesario, los papeles se indicarán mediante etiquetas situadas sobre las líneas que unen los conjuntos de entidades con los de relaciones para señalar su función.
- La relación de cardinalidad se representa mediante una flecha en la parte de “varios”. De esta forma, las representaciones serían:



## 6. CONJUNTOS DE ENTIDADES DÉBILES.

### 6.1. DEFINICIÓN.

Un ejemplo de conjunto de entidades débiles sería el siguiente:



## 6.2. CARACTERÍSTICAS.

Características del anterior esquema:

- La relación es de uno a varios y no tiene atributos descriptivos.
- Aparece una entidad dominante (préstamo) y una subordinada (pago): dependencia en clave.
- Existe un **discriminante** o **clave parcial** que permite diferenciar una entidad del conjunto de entidades débiles del resto. En este ejemplo la clave parcial de un pago sería el número de pago.

De forma general, podemos obtener la clave primaria de un conjunto de entidades débiles como la unión de su clave parcial y de la clave primaria del conjunto de entidades fuertes al que está subordinado.

Al conjunto de entidades fuerte también se le denomina **propietario** del conjunto de entidades débil al que subordina.

Una **relación de identificación** es la relación que existe entre el propietario y el conjunto de entidades débil. Esta relación se representa mediante un rombo doble.





# TEMA 3.      MODELOS

## E/R EXTENDIDO Y RELACIONAL

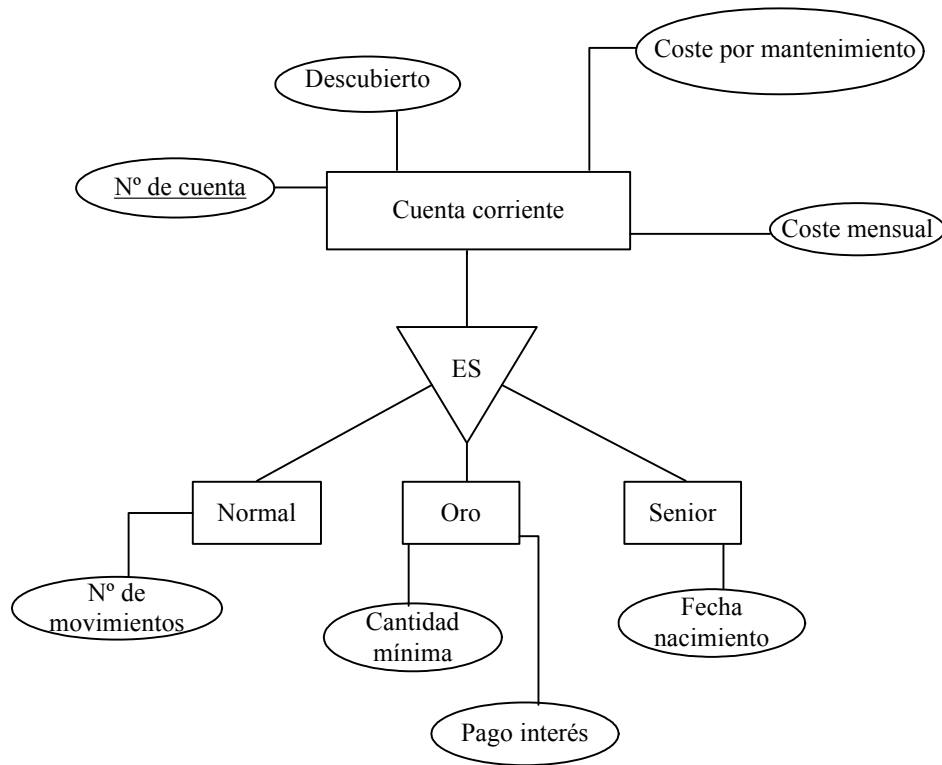
### **1. MODELO ENTIDAD/RELACIÓN EXTENDIDO.**

#### **1.1. ESPECIALIZACIÓN.**

La especialización consiste en el proceso de designación de un subgrupo dentro de un determinado conjunto de entidades. Estos subgrupos serán designados atendiendo a los atributos específicos de cada subgrupo.

La relación básica que determina una especialización es la relación “ES”; ésta es una relación de superclase-subclase porque relaciona una clase padre con una clase hijo.

Ejemplo: algunas entidades del conjunto de entidades “Cuenta corriente” tienen atributos específicos. Dicho conjunto de entidades se puede especializar en tres subgrupos: cuentas normales, cuentas oro y cuentas senior, cada uno de ellos con sus atributos específicos. Por otra parte, el conjunto de entidades “Cuenta corriente” tiene unos atributos genéricos, que aparecen en todos los subgrupos: número de cuenta, descubierto, coste por mantenimiento y coste mensual.



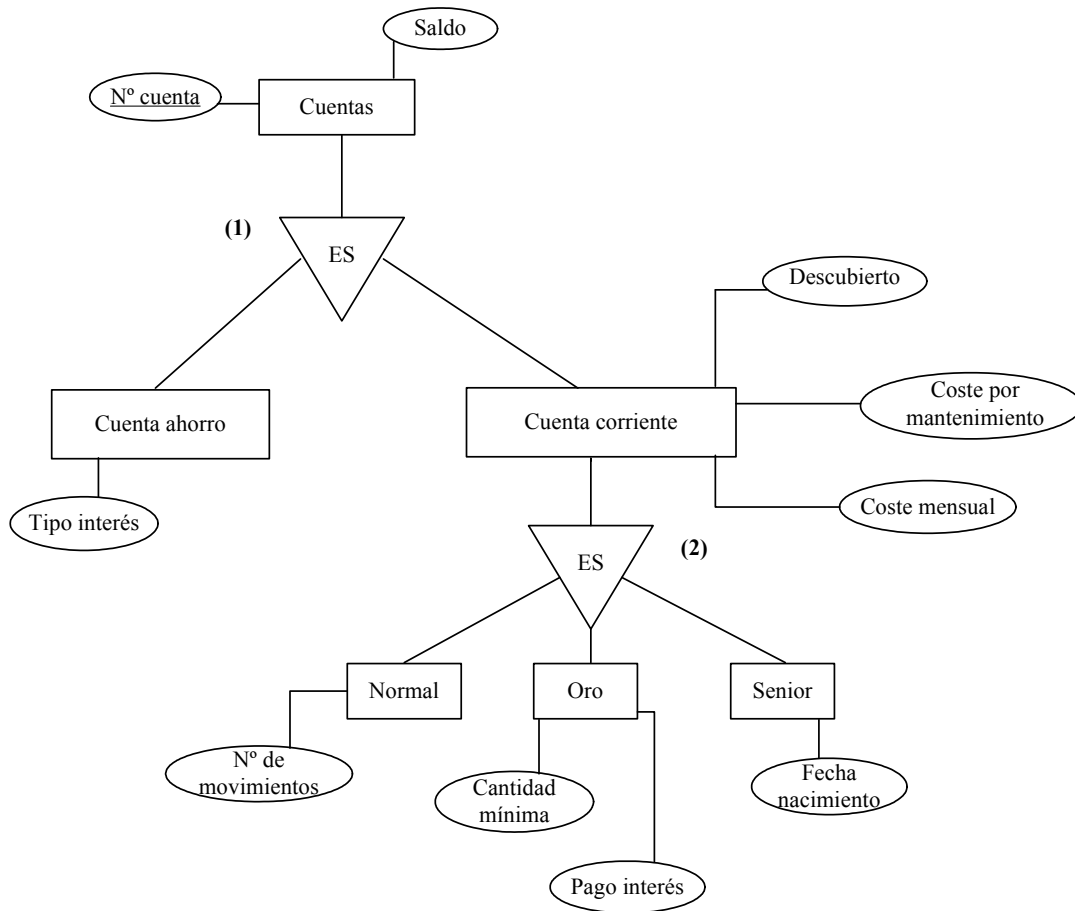
## 1.2. GENERALIZACIÓN.

La generalización es una relación que permite expresar similitud entre un conjunto de entidades de nivel más bajo, sintetizando un conjunto de entidades de un nivel más alto a partir de atributos comunes.

La generalización se obtiene mediante diseño ascendente; esto permite resaltar similitudes y economizar la representación ya que no se repiten atributos.

Las diferencias fundamentales entre la especialización y la generalización son principalmente dos:

- El punto de partida: la especialización se obtiene mediante un diseño descendente mientras que la generalización se obtiene mediante un diseño ascendente.
- El objetivo global: el objetivo de la especialización es diferenciar entidades, el de la generalización es sintetizarlas.



(1) ≡ Generalización.

(2) ≡ Especialización.

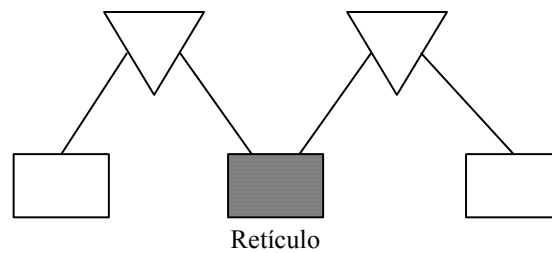
### 1.3. HERENCIA.

Tanto en la generalización como en la especialización se produce herencia de atributos.

Características de la herencia:

- Los conjuntos de entidades de niveles inferiores heredan los atributos de los conjuntos de entidades superiores.
- La herencia se aplica a través de todas las relaciones.
- Existirá una jerarquía de conjuntos de entidades cuando se tengan más de dos niveles de relaciones “ES”.

Se denomina **retículo** de una jerarquía al conjunto de entidades de nivel más inferior que participa en más de un conjunto de relaciones de tipo “ES”.

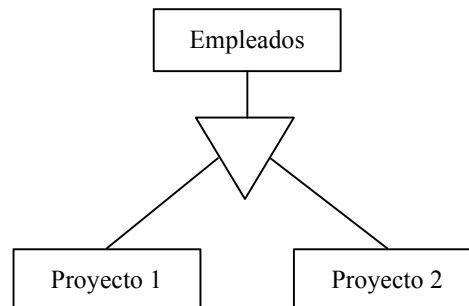


#### 1.4. LIGADURAS DE DISEÑO EN ESPECIALIZACIÓN.

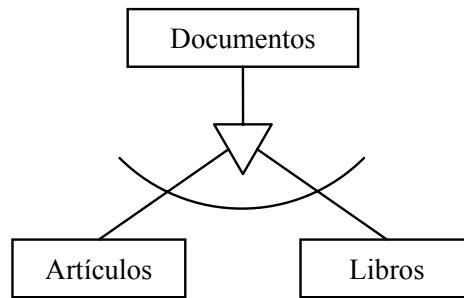
Las ligaduras de diseño indican las relaciones existentes entre entidades padre y entidades hija. Estas relaciones suelen ser especializaciones o generalizaciones que benefician en gran medida al diseño de la solución del problema.

Algunos puntos importantes a determinar en las ligaduras de diseño son los siguientes:

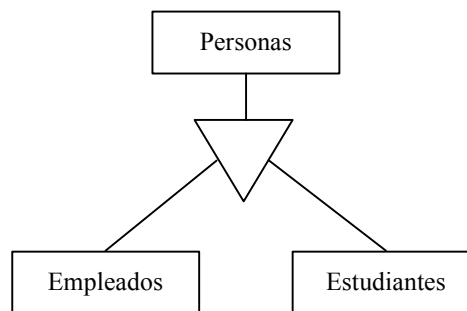
- a) **Determinar pertenencia:** se debe determinar si el conjunto de entidades padre es de un determinado conjunto de entidades hija. Esta pertenencia puede ser determinada mediante los siguientes métodos:
  - i. *Definida por condición:* para que la entidad padre pertenezca a una entidad hija, debe cumplir una determinada condición o predicado. Por ejemplo, las generalizaciones quedan definidas por un atributo denominado atributo discriminante.
  - ii. *Definida por el usuario:* el usuario determina la pertenencia o no de la entidad padre en una entidad hija. Por ejemplo, el usuario decide a qué proyecto está asignado cada empleado:



- b) **Cuantificar la pertenencia:** se debe determinar si un conjunto de entidades de nivel alto puede pertenecer a más de un conjunto de entidades de nivel más bajo. Atendiendo a esta cuantificación se pueden definir los siguientes términos:
  - i. *Conjunto de entidades disjunto:* un padre pertenece a un hijo o a otro, no puede pertenecer a dos hijos a la vez. Este tipo de ligadura debe ser explicitada, es decir, debe quedar representada en el diagrama entidad-relación. Por ejemplo, dado un documento, puede ser un artículo o un libro, no ambas cosas a la vez.



- ii. *Solapado*: una entidad padre puede pertenecer a más de una entidad hija. Por ejemplo, las personas pueden ser tanto empleados como estudiantes, como ambas cosas a la vez:



- c) **Ligadura de completitud**: especifica si un conjunto de entidades de un nivel más alto tiene o no que pertenecer a un conjunto de entidades de un nivel más bajo. La ligadura de completitud puede ser de dos tipos:
- i. *Total*: cada entidad padre debe pertenecer al menos a una entidad hijo.
  - ii. *Parcial*: cada entidad padre no tiene porqué pertenecer a una entidad hijo.







### 1.5. DIAGRAMA ENTIDAD/RELACIÓN EXTENDIDO.

- a) Conjunto de entidades.

- Fuerte:
- Débil:

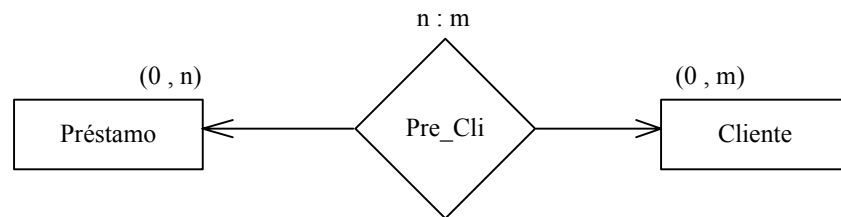
- b) Atributos.

- Normal:
- Identificador principal (clave primaria):
- Identificador alternativo:

- Identificador compuesto: 
- Multivalorado: 
- Atributo compuesto: 
- Atributo multivalorado compuesto: 
- Atributo opcional: 
- Atributo derivado:  Lleva una etiqueta además del nombre y suele estar relacionada con la función de derivación que se utiliza para hallarlo.

c) Cardinalidad de las entidades: se indica por un número mínimo y un número máximo. Por ejemplo:

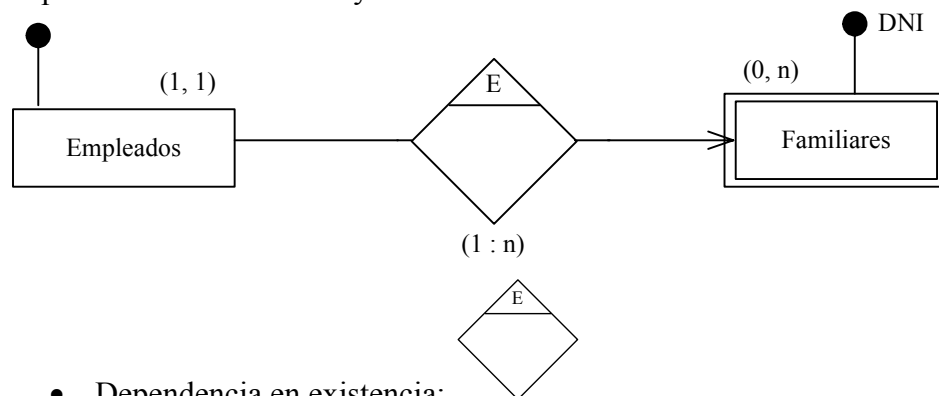
- Caso 1:  $R ( E_1 (0, n) : E_2 (0, m) )$



- Caso 2:  $R ( E_1 (0, n) : E_2 (1, n) )$

NOTA: En el diagrama E/R Extendido se cambia el significado de la flecha. Cuando la cardinalidad máxima es “uno”, no se pone flecha. La flecha se dibujará en la entidad correspondiente a la parte “varios” (al contrario de lo explicado en el diagrama E/R básico). Se mantienen ambas nomenclaturas para que se conozcan, pero se utilizará por defecto la del diagrama E/R Extendido.

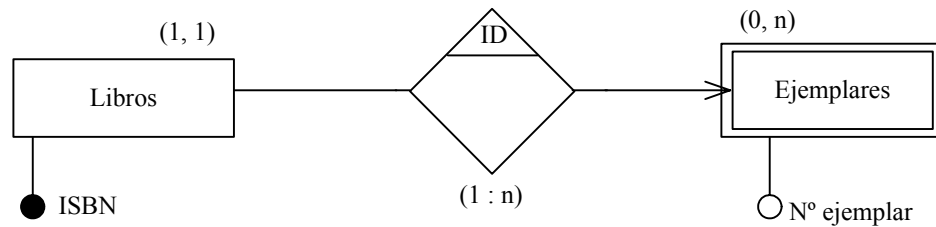
d) Dependencia en existencia y en identificación.



- Dependencia en existencia:

- Dependencia en identificación:

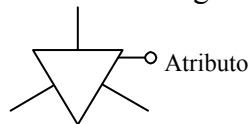
Ejemplo:



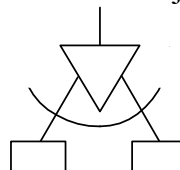
Cuando se tiene una dependencia en identificación siempre va a haber también una dependencia en existencia, pero esto no se da a la inversa.

e) Generalización y herencia.

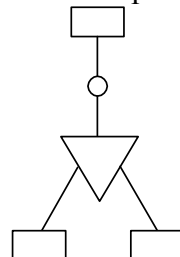
- 1) Determinar pertenencia: cuando venga definida por un atributo.



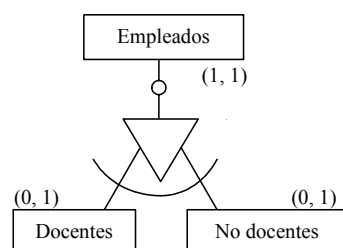
- 2) Cuantificación: representación de conjuntos de entidades disjuntos.



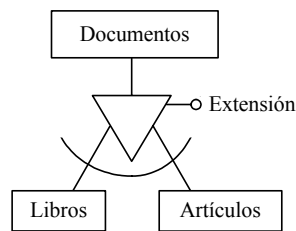
- 3) Obligatoriedad: representación de participación total.



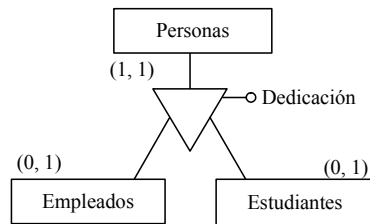
Ejemplos:



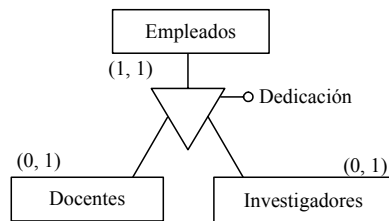
- 1) Un empleado es o bien docente o bien no docente (uno de los dos obligatoriamente). La pertenencia la determina el usuario, no un atributo.



- 2) Según la extensión se determina si un documento es libro o artículo. Un artículo no puede ser un libro y viceversa (disjunto). Hay más tipos de documentos (no es total): revistas, etc.



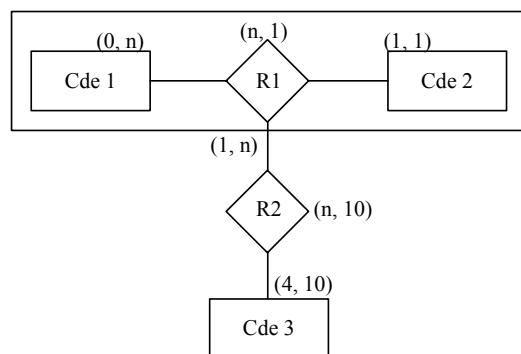
- 3) El conjunto de entidades "Personas" es solapado: pueden ser empleados y estudiantes a la vez. No es total sino parcial: no todo el mundo tiene que ser una cosa u otra (puede haber jubilados, etc.). La pertenencia viene definida por el atributo "Dedicación".



- 4) "Empleados" no es disjunto porque pueden ser ambas cosas a la vez (solapado). Como puede haber más tipos de empleados, es parcial y el atributo "Dedicación" determina la participación.

## 1.6. AGREGACIÓN.

La agregación es una abstracción mediante la cual las relaciones se tratan igual que las entidades, es decir, la agregación se utiliza para expresar relaciones entre conjuntos de relaciones. Dada una relación entre dos conjuntos de entidades se quiere que otro conjunto de entidades participe en esa relación:



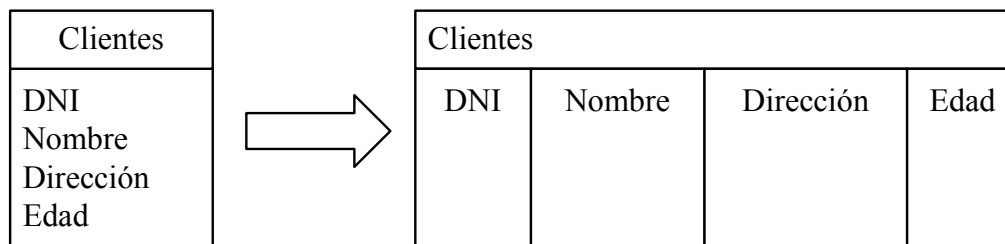
El hacer agregaciones es muy similar a implementar relaciones ternarias, sin embargo, las agregaciones aportan más información dado que en una agregación se pueden explicitar todas las relaciones, cosa que en una relación ternaria no se puede hacer. En general, una agregación refleja más semántica.



## 2. REDUCCIÓN DE UN ESQUEMA E/R A RELACIONAL.

### 2.1. REDUCCIÓN DE UN CONJUNTO DE ENTIDADES FUERTE.

Sea E un conjunto de entidades fuerte con atributos  $\{a_1, a_2, \dots, a_n\}$  se representará mediante una tabla denominada E que contenga 'n' columnas distintas cada una de ellas correspondiente a un atributo de E.



Cada fila de esta tabla (tupla) es un ejemplar del conjunto de entidades, una entidad particular.

### 2.2. REDUCCIÓN DE UN CONJUNTO DE ENTIDADES DÉBILES.

Sea A un conjunto de entidades débil con atributos  $\{a_1, a_2, \dots, a_n\}$  y sea B el conjunto de entidades fuerte del que depende con atributos de clave primaria  $\{b_1, b_2, \dots, b_m\}$ .

El conjunto de entidades débil A se representará mediante una tabla también denominada A con una columna para cada uno de los atributos del conjunto formado por la unión de los atributos de A y B.

### 2.3. REDUCCIÓN DE UN CONJUNTO DE RELACIONES.

#### 2.3.1. Caso general.

Sea el conjunto de relaciones R y sea  $\{a_1, a_2, \dots, a_n\}$  el conjunto de los atributos formados por la unión de las claves primarias de los conjuntos de entidades que participan en R. Sea  $\{b_1, b_2, \dots, b_m\}$  el conjunto de los atributos asociados a R.

Este conjunto de relaciones se representa mediante una tabla también denominada R que tendrá como columnas las asociadas a los atributos del conjunto formada por la unión de  $\{a_1, a_2, \dots, a_n\}$  y  $\{b_1, b_2, \dots, b_m\}$ .

#### 2.3.2. Redundancia de tablas.

En general, la tabla para el conjunto de relaciones que une un conjunto de entidades débil con un conjunto de entidades fuerte es redundante, con lo que no tiene que ser obligatoriamente representada.

#### 2.3.3. Combinación de tablas.

Cuando se tiene un conjunto de relaciones varios a uno del conjunto de entidades A en el conjunto de entidades B, en general, habrá que manejar tres tablas: la A, la B y la procedente del conjunto de relaciones (AB). Sin embargo, cuando haya una dependencia de existencia del conjunto de entidades A al conjunto de entidades B se

pueden utilizar sólo dos tablas: la B y la C, donde esta última se obtiene como combinación de las tablas A y AB, es decir,  $C = A + AB$ .

#### 2.4. REDUCCIÓN DE LOS ATRIBUTOS MULTIVALORADOS.

Para un atributo multivalorado M se creará una tabla T con una columna que corresponda con la clave primaria del conjunto de entidades o de relaciones del cual M es atributo, y otra columna para cada una de las distintas posibilidades del atributo multivalorado.

#### 2.5. REDUCCIÓN DE UNA GENERALIZACIÓN.

Existen dos métodos que se pueden aplicar para obtener la reducción de tablas:

1. Método utilizado para el caso general: se crea una tabla para el conjunto de entidades más alto (padre) y otra para cada conjunto de entidades de nivel más bajo (hijos). En la tabla del hijo se debe incluir una columna para cada uno de sus atributos y una columna para cada uno de los atributos de la clave primaria del padre.
2. Método utilizado cuando se trabaje con una generalización completa y disjunta: en este caso se crea una tabla para cada una de las entidades hijo con una columna para cada uno de sus atributos y otra columna para cada uno de los atributos del padre.

#### 2.6. REDUCCIÓN DE UNA AGREGACIÓN.

Se creará una tabla para el conjunto de relaciones que asocian el conjunto de entidades y la agregación. En esta tabla se debe incluir una columna para cada atributo de la clave primaria del conjunto de entidades y del conjunto de relaciones que conforman la agregación. En caso de que existan, también se debe incluir una columna para cada uno de los atributos descriptivos del conjunto de relaciones que une el conjunto de entidades con la agregación.

#### *Resumen de la reducción a tablas:*

| Tipo de reducción             | Número de tablas  |
|-------------------------------|---|
| Conjunto de entidades fuerte  | 1   |
| Conjunto de entidades débiles | 1   |
| Conjunto de relaciones        | El número depende de la redundancia existente y la combinación de tablas que se pueda realizar. |
| Atributos multivalorados      | 1   |
| Generalización                | Caso general : 1 tabla para el padre y otra para cada hijo.                                     |
|                               | Generalización disjunta y completa: si interesa, 1 tabla para cada hijo.                        |

| Tipo de reducción | Número de tablas  |
|-------------------|---|
| Agregación        | 1 tabla para el conjunto de relaciones de la agregación cuya clave primaria será la del conjunto de entidades junto con la del conjunto de relaciones de la agregación (claves primarias de los dos conjuntos de entidades de la relación). |

### 3. CONSIDERACIONES ADICIONALES.

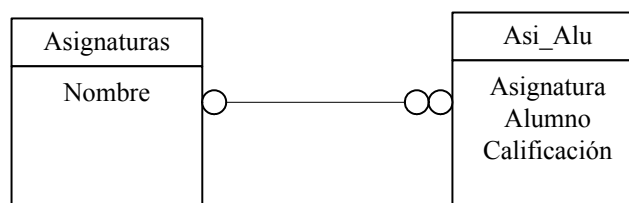
#### 3.1. RESTRICCIONES INHERENTES AL MODELO RELACIONAL.

- En una tabla no pueden existir dos tuplas iguales, es decir, todas las filas de una tabla tienen que ser distintas. Para ello se necesita una clave primaria única.
- El orden de las tuplas no tiene importancia.
- Cada atributo toma un único valor dentro de su dominio.
- Ningún atributo perteneciente a la clave primaria puede tomar valor NULL.

#### 3.2. RESTRICCIONES SEMÁNTICAS.

- **Restricciones de clave primaria (PRIMARY KEY):** permiten declarar un atributo o un conjunto de atributos como clave primaria.
- **Restricciones de unicidad (UNIQUE):** permiten declarar claves alternativas. Se debe tener en cuenta que todas estas claves alternativas tienen que ser distintas.
- **Restricciones de obligatoriedad (NOT NULL):** permiten determinar aquellos atributos que deben tomar obligatoriamente algún valor.
- **Restricciones de clave ajena (FOREIGN KEY):** estas claves ajenas se utilizan para enlazar tablas dentro de una base de datos siempre manteniendo la integridad referencial.

Se define una **clave ajena** como aquel atributo de una tabla que es clave primaria en otra tabla con la cual está relacionada. Esta clave ajena sólo puede tomar valores que estén permitidos en la clave primaria, aunque no tienen por qué tener obligatoriamente el mismo nombre. En el ejemplo siguiente, “Nombre” sería clave primaria y “Asignatura”, clave externa o ajena.



### 3.3. OPCIONES DE BORRADO Y MODIFICACIÓN EN LAS CLAVES AJENAS.

Estas opciones indican las acciones a realizar en la clave ajena (de la entidad hijo) cuando se borra la entidad padre (consecuentemente también se borra la clave primaria del padre). Las opciones posibles son las siguientes:

- **Borrado en cascada (CASCADE):** cuando se borra al padre se borran todos los hijos (siempre se habla del borrado de tuplas).
- **Borrado restringido (RESTRICT):** no se puede borrar al padre en caso de que tenga algún hijo.
- **Borrado con puesta a valores nulos (SET NULLS):** siempre que la clave ajena lo permita, al borrar al padre se pone la clave ajena a valor nulo.
- **Borrado con puesta a valor por defecto (SET DEFAULT):** al borrar al padre la clave ajena del hijo toma un valor por defecto que ya ha sido determinado con anterioridad a crear la relación.

### 3.4. OTROS MECANISMOS PARA REPRESENTAR LAS RESTRICCIONES.

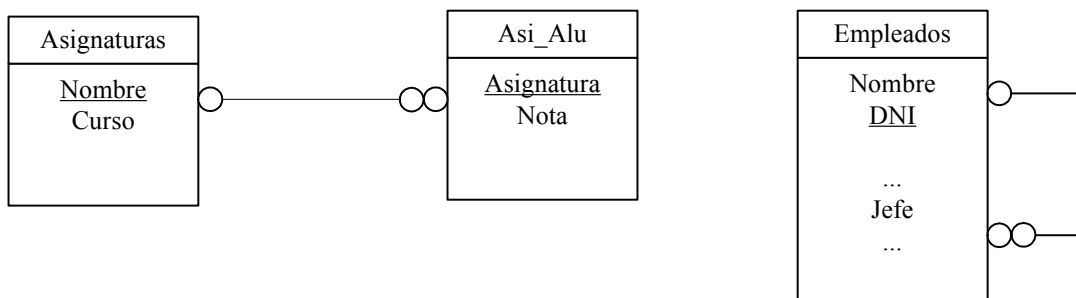
- **Restricción de verificación (CHECK):** con esta restricción se especifica una condición para los valores de un atributo. Estas restricciones no necesitan ningún nombre; solamente hace falta definirlas.
- **Aserciones (ASSERTION):** especifican condiciones de valores de atributos en varias tablas. Se debe poner un nombre específico a cada una de las aserciones dado que no se sabe a qué tablas se aplican exactamente.
- **Disparadores (TRIGGERS):** especifican una acción (siempre distinta al rechazo) cuando no se cumpla una determinada restricción semántica. Un ejemplo serían las condiciones aplicadas en la inserción en tablas: al insertar un empleado nuevo en la tabla “Trabajadores” (más general) se quiere que lo inserte también en la de “Vendedores” (más específica). Se pueden considerar reglas ‘ECA’ (evento, condición, acción), según las cuales, dado un evento y cumpliendo una condición, se realiza una determinada acción.

### 3.5. ALGUNAS CONSIDERACIONES SOBRE LA NOTACIÓN UTILIZADA.

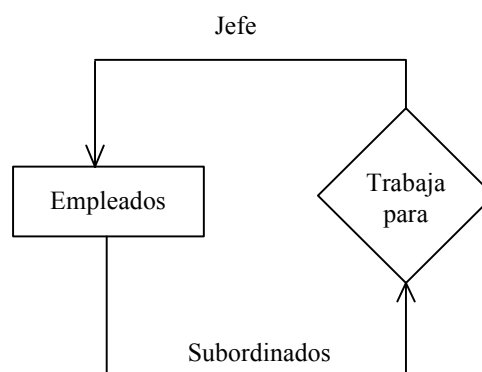
- El nombre de la tabla se pondrá en mayúsculas.
- El nombre de la clave primaria debe estar subrayado.
- Cualquier valor asignado a una clave ajena debe pertenecer a los valores permitidos por la clave primaria a la que está asociada.
- Aquellos atributos que no pueden tomar valor NULL se marcan con un asterisco (\*).
- Aquellos atributos cuyo valor NOT NULL venga impuesto por la reducción del diagrama Entidad/Relación a Relacional o como una

restricción indicada en el enunciado del problema, se marcarán con un asterisco en el interior de un círculo. ⊛

- El lado en el que se encuentra la clave primaria se determinará con un círculo.
- El lado en el que se encuentra la clave ajena se denotará con un doble círculo.
- El borrado en cascada se denotará como B:C.
- El borrado restringido se denotará como B:R.
- El borrado con puesta a NULL se denotará como B:N.
- El borrado con puesta a valores por defecto se denotará como B:D.
- La modificación en cascada se denotará como M:C.
- La modificación restringida se denotará como M:R.
- La modificación con puesta a NULL se denotará como M:N.
- La modificación con puesta a valores por defecto se denotará como M:D.



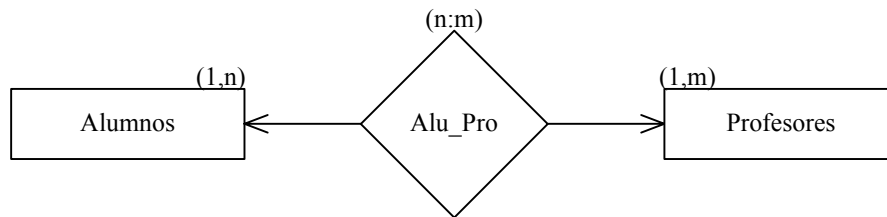
Las claves ajenas también pueden tomar valores que procedan de la misma tabla, como ocurriría en el modelo de la derecha, que corresponde al siguiente diagrama E/R:



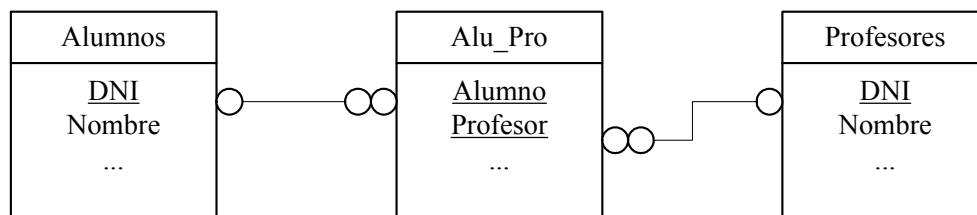
## 4. EL PROBLEMA DEL ESTUDIANTE.

Se tienen tres situaciones:

CASO A: Un alumno puede tener varios profesores y un profesor puede tener varios alumnos.

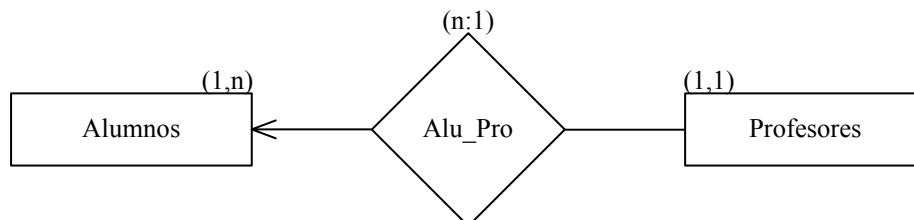


**Reducción a tablas:**

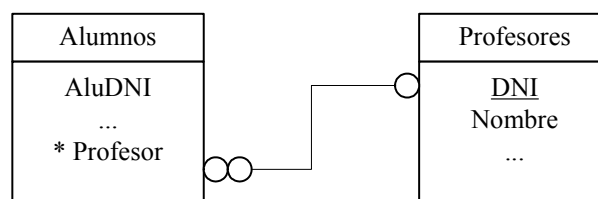


El nombre del campo de la clave ajena y de la clave primaria pueden ser diferentes, pero deben tener el mismo tipo de datos.

CASO B: Un alumno puede tener sólo un profesor y un profesor puede tener varios alumnos.

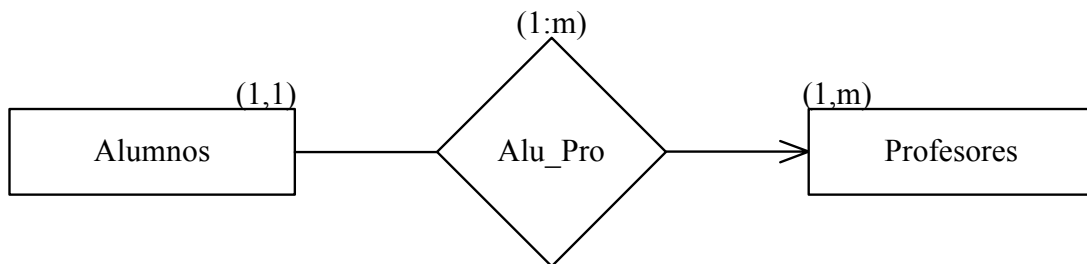


**Reducción a tablas:**

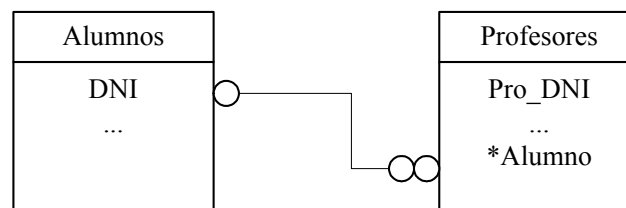


Como en la tabla auxiliar queda una relación uno a uno, no se necesita la tabla de la relación "Alu\_Pro".

CASO C: Un alumno puede tener varios profesores y un profesor sólo puede dar clase a un alumno.



### Reducción a tablas:



Como en la tabla auxiliar aparece una relación uno a uno no se necesita la tabla de la relación “Alu\_Pro”.

## 5. PROBLEMAS DEL MODELO ENTIDAD/RELACIÓN.

- Se desea diseñar un esquema relacional de una base de datos para un centro de enseñanza que contenga información sobre los alumnos, las asignaturas y las calificaciones que se obtienen en cada una de las mismas.  
Desarrollar un modelo E/R del mismo y posteriormente reducirlo a tablas.  
Añada los campos y la información necesaria para que el diseño sea correcto.

- Se desea diseñar una base de datos para una Universidad que contenga información sobre los alumnos, las asignaturas y las carreras que se pueden estudiar. Construir un modelo E/R y pasarlo posteriormente a un esquema relacional teniendo en cuenta las siguientes restricciones:

- Un alumno puede estar matriculado en muchas asignaturas.
- Una asignatura sólo puede pertenecer a una sola carrera.
- Una carrera puede tener muchas asignaturas.
- Es necesario reflejar las calificaciones.

Añada los campos y la información necesaria para que el diseño sea correcto.

- Se desea diseñar una base de datos para una Universidad que contenga información sobre los alumnos, las asignaturas y los profesores. Construir un modelo E/R y pasarlo posteriormente a un esquema relacional teniendo en cuenta las siguientes restricciones:
- Una asignatura puede estar impartida por muchos profesores (no a la vez) ya que pueden existir grupos.
  - Un profesor puede dar clase de muchas asignaturas.

- Un alumno puede estar matriculado en muchas asignaturas.
- Se necesita tener constancia de las asignaturas en las que está matriculado un alumno, la nota obtenida y el profesor que le ha calificado.
- También es necesario tener constancia de las asignaturas que imparten todos los profesores (independientemente de si tienen algún alumno matriculado en su grupo).
- No existen asignaturas con el mismo nombre.
- Un alumno no puede estar matriculado en la misma asignatura con dos profesores distintos.

Añada los campos y la información necesaria para que el diseño sea correcto.

4. Se desea diseñar una base de datos para una sucursal bancaria que contenga información sobre los clientes, las cuentas, las sucursales y las transacciones producidas. Construir un modelo E/R y pasarlo posteriormente a un esquema relacional teniendo en cuenta las siguientes restricciones:

- Una transacción viene determinada por su número de transacción, la fecha y la cantidad.
- Una transacción es movimiento sobre una cuenta, pudiendo deberse a domiciliaciones, abonos o cargos no realizados necesariamente por clientes del banco.
- Un cliente puede tener muchas cuentas.
- Una cuenta puede pertenecer a varios clientes.
- Una cuenta solamente puede estar en una sucursal.

Añada los campos y la información necesaria para que el diseño sea correcto.

5. Se desea diseñar una base de datos para un centro comercial organizado por departamentos que contenga información sobre los clientes que han comprado algo, los trabajadores, el género que se oferta y las ventas realizadas. Construir un modelo E/R y pasarlo posteriormente a un esquema relacional, teniendo en cuenta las siguientes restricciones:

- Existen tres tipos de trabajadores; gerentes, jefes y vendedores.
- Cada departamento está gobernado por un gerente.
- Un determinado producto sólo se encuentra en un departamento.
- Los jefes y vendedores sólo pueden pertenecer a un único departamento.
- Un gerente tiene a su cargo a un cierto número de jefes y éstos a su vez a un cierto número de vendedores.
- Una venta la realiza un vendedor a un cliente y debe quedar constancia del artículo vendido. Sólo un artículo por apunte de venta.

Añada los campos y la información necesaria para que el diseño sea correcto.

6. Se desea diseñar una base de datos para una discoteca-videoteca que contenga información de videos, discos, socios, empleados y préstamos. Construir un modelo E/R y pasarlo posteriormente a un esquema relacional teniendo en cuenta las siguientes restricciones:



- Un socio puede tener en préstamo varios videos y discos a la vez.
- Un video o disco sólo puede estar prestado a un socio.
- Un empleado puede prestar muchos discos y videos.
- Cuando se realiza un préstamo debe aparecer el socio, el video o disco, la fecha y el empleado.
- En los discos debe aparecer información sobre su autor y en los vídeos, su protagonista.

Añada los campos y la información necesaria para que el diseño sea correcto.

7. Se desea diseñar una base de datos para una agencia matrimonial que contenga información de hombres (con todos su datos personales), mujeres (con todos sus datos personales), empleados (divididos en tres categorías: socios, directores y administrativos), citas realizadas (debe quedar constancia de la fecha, el hombre, la mujer y el director que la promovió) y matrimonios (fecha, hombre y mujer). Construir un modelo E/R y pasarlo posteriormente a un esquema relacional teniendo en cuenta las siguientes restricciones:

- Un hombre puede tener cita con varias mujeres.
- Una mujer puede tener cita con varios hombres.
- Un hombre puede casarse con varias mujeres (puede enviudar).
- Una mujer puede casarse con varios hombres (puede enviudar).
- Solo los directores pueden promover citas.
- Un socio tiene a su cargo a varios directores y éstos, a su vez, a varios administrativos.

8. Se desea diseñar una base de datos que contenga información sobre municipios, viviendas y personas. Cada persona sólo puede habitar en una vivienda y estar empadronada en un municipio, pero puede ser propietaria de varias viviendas. Una vivienda pertenece a un único municipio. Interesa conocer qué personas dependen del cabeza de familia (C.F.). Realizar el modelo E/R y posteriormente el relacional.

9. Realizar un modelo E/R del siguiente problema, indicando:

- Todos los atributos de las entidades y de las relaciones (si los tuvieran), y los atributos que son clave primaria.
- Señalar igualmente si el modelo propuesto no recoge toda la semántica del problema, indicando, en este caso, las restricciones necesarias para hacerlo.
- Si aparece especialización o generalización, especificar el tipo.

Se pretende llevar a cabo un control sobre la energía eléctrica que se produce y consume en un determinado país. Se parte de las siguientes hipótesis:

- Existen productores básicos de electricidad que se identifican por un nombre, de los cuales interesa su producción media, producción máxima y fecha de entrada en funcionamiento. Estos productores básicos lo son de una de las siguientes categorías: Hidroeléctrica, Solar, Nuclear o Térmica.

De una central hidroeléctrica o presa interesa saber su ocupación, capacidad máxima y número de turbinas. De una central solar, interesa saber la superficie total de paneles solares, la media anual de horas de sol y tipo (fotovoltaica o termodinámica). De una central nuclear, interesa saber el número de reactores que posee, el volumen de plutonio consumido y el de residuos nucleares que produce. De una central térmica interesa saber el número de hornos que posee, el volumen de carbón consumido y el volumen de su emisión de gases.

- Por motivos de seguridad nacional, interesa conocer el plutonio de que se provee una central nuclear. Este control se refiere a la cantidad de plutonio que compra a cada uno de sus posibles suministradores (nombre y país) y que porta un determinado transportista (nombre y matrícula). Ha de tenerse en cuenta que el mismo suministrador puede vender plutonio a distintas centrales nucleares y que cada porte, (un único porte por compra), puede realizarlo un transportista diferente.
- Cada día, los productores entregan la energía producida a una o varias estaciones primarias, las cuales pueden recibir diariamente una cantidad distinta de energía de cada uno de estos productores. Los productores entregan siempre el total de su producción. Las estaciones primarias se identifican por su nombre y tienen un número de transformadores de baja a alta tensión, siendo además cabecera de una o varias redes de distribución.
- Una red de distribución se identifica por un número de red y sólo puede tener una estación primaria como cabecera. La propiedad de una red puede ser compartida por varias compañías eléctricas; a cada compañía eléctrica se le identifica por su nombre.
- La energía sobrante en una de las redes puede enviarse a otra red. Se registra el volumen total de energía intercambiada entre dos redes.
- Una red está compuesta por una serie de líneas; cada línea se identifica por un número secuencial dentro del número de red y tiene una determinada longitud. La menor de las líneas posibles abastecerá al menos a dos subestaciones.
- Una estación es abastecida sólo por una línea y distribuye a una o varias zonas de servicio. A estos efectos, las provincias (código y nombre), se encuentran divididas en estas zonas de servicio, aunque no puede haber zonas de servicio que pertenezcan a más de una provincia. Cada zona de servicio puede ser atendida por más de una subestación.
- En cada zona de servicio se desea registrar el consumo medio y el número de consumidores finales de cada una de las siguientes categorías: particulares, empresas e instituciones.

10. Realizar un diagrama E/R (extendido) del siguiente problema, indicando:

- Todos los atributos de las entidades y de las relaciones (si los tuvieran), y los atributos que son clave primaria.

- Señalar igualmente si el diagrama propuesto no recoge toda la semántica del problema, indicando en este caso las restricciones necesarias para hacerlo.
- Si aparece especialización o generalización, indicar el tipo.

El club de Ajedrez de la Universidad de León ha sido encargado por la Federación Internacional de Ajedrez de la organización de los próximos campeonatos internacionales infantiles que se celebrarán en esta localidad. Debido a esto, desea llevar a una base de datos toda la gestión relativa a participantes, alojamientos y partidas. Para ello hay que tener en cuenta lo siguiente:

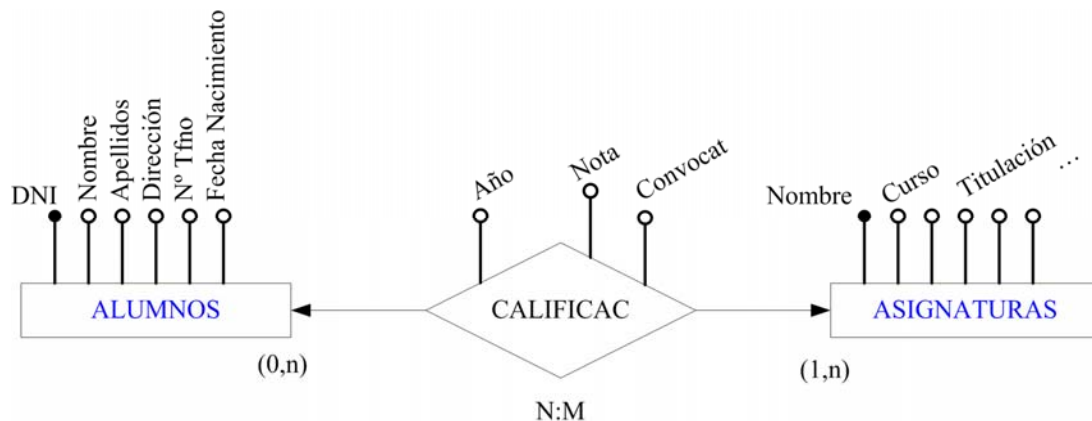
- En el campeonato participan jugadores y árbitros. De ambos se requiere conocer el número de asociado, nombre, dirección, teléfono de contacto y campeonatos en los que han participado (como jugador o como árbitro). De los jugadores se precisa además el nivel de juego en una escala de 1 a 10.
- Ningún árbitro puede participar como jugador.
- Los países envían al campeonato un conjunto de jugadores y árbitros, aunque no todos los países envían participantes. Todo jugador y árbitro es enviado por un único país. Un país puede ser representado por otro país.
- Cada país se identifica por un número correlativo según su orden alfabético e interesa conocer además de su nombre, el número de clubes de ajedrez existentes en el mismo.
- Cada partida se identifica por un número correlativo (CodPar), la juegan dos jugadores y la arbitra un árbitro. Interesa registrar las partidas que juega cada jugador y el color (blancas o negras) con el que juega. Ha de tenerse en cuenta que un árbitro no puede arbitrar a jugadores enviados por el mismo país que le ha enviado a él.
- Todo participante compite en al menos una partida.
- Tanto jugadores como árbitros se alojan en uno de los hoteles en los que se desarrollan las partidas, se desea conocer en qué hotel y en qué fechas se ha alojado cada uno de los participantes. Los participantes pueden no permanecer en León durante todo el campeonato, siendo posible que acudan únicamente cuando tienen que jugar alguna partida alojándose en el mismo o distinto hotel. De cada hotel, se desea conocer el nombre, la dirección y el número de teléfono.
- El campeonato se desarrolla a lo largo de una serie de jornadas (año, mes, día) y cada partida tiene lugar en una de las jornadas aunque no tengan lugar partidas todas las jornadas.
- Cada partida se celebra en una de las salas de las que pueden disponer los hoteles; se desea conocer el número de entradas vendidas en la sala para cada partida. De cada sala, se desea conocer la capacidad y medios de que dispone (radio, televisión, vídeo,...) para facilitar la retransmisión de los encuentros. Una sala puede disponer de varios medios distintos.

- De cada partida se pretenden registrar todos los movimientos que la componen; la identificación de movimiento se establece en base a un número de orden dentro de cada partida: para cada movimiento se guarda la jugada (5 posiciones) y un breve comentario realizado por un experto.

## 6. SOLUCIONES PROPUESTAS A LOS EJERCICIOS DE E/R.

### EJERCICIO 1.

#### Esquema Entidad Relación.

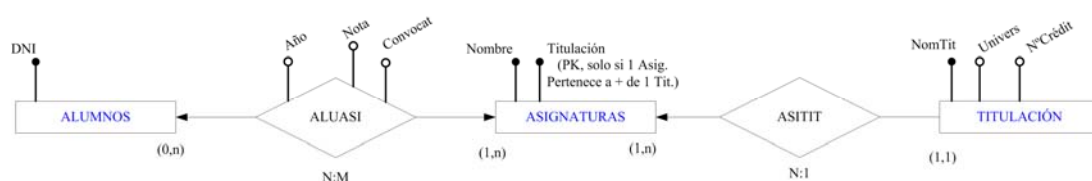


#### Esquema relacional.

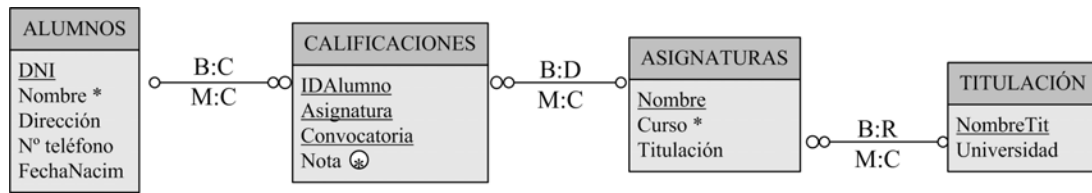


### EJERCICIO 2.

#### Esquema Entidad Relación.

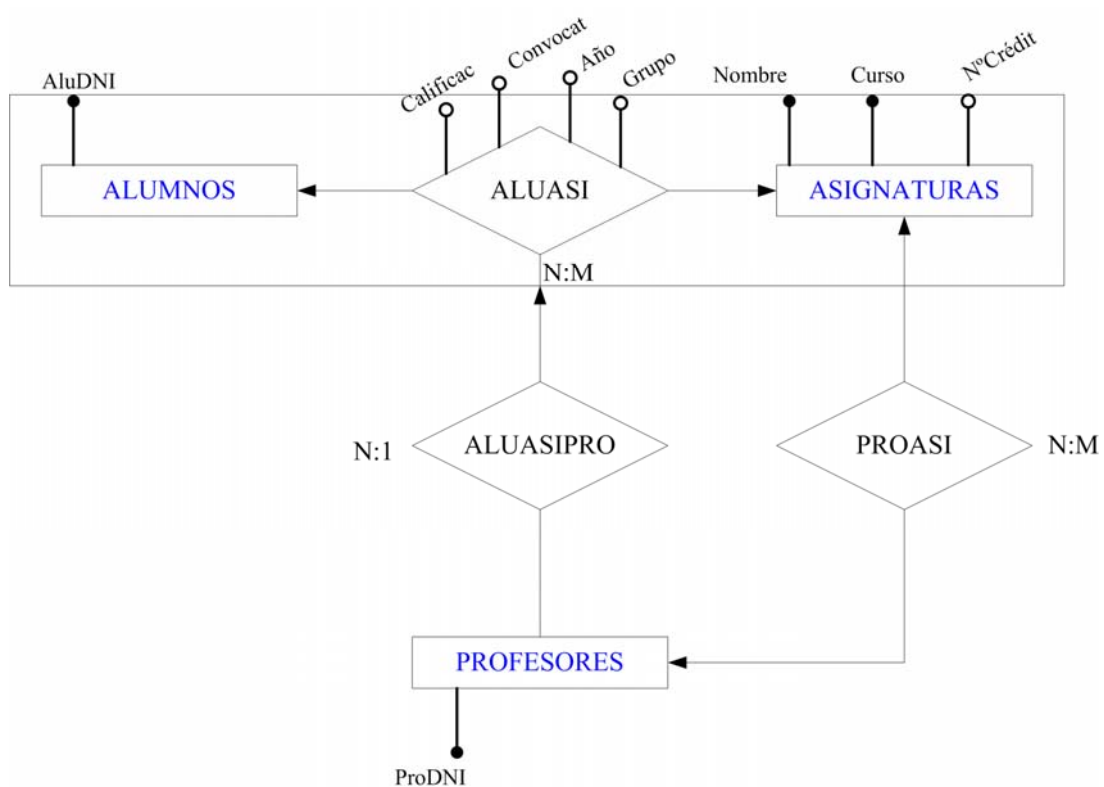


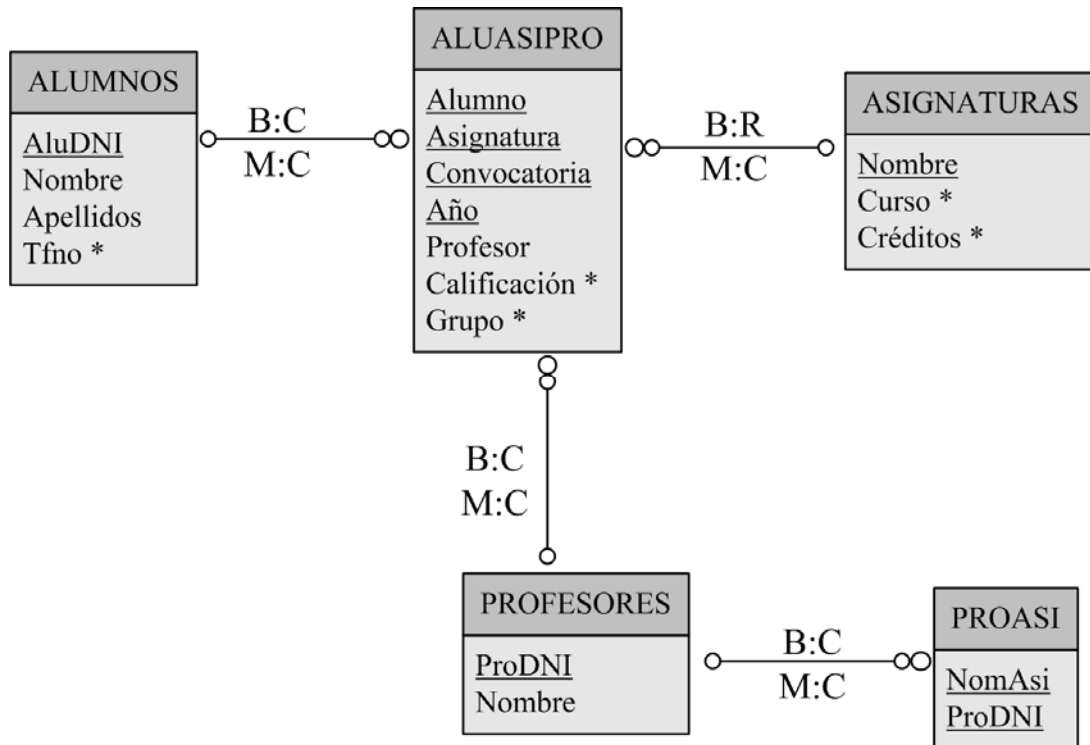
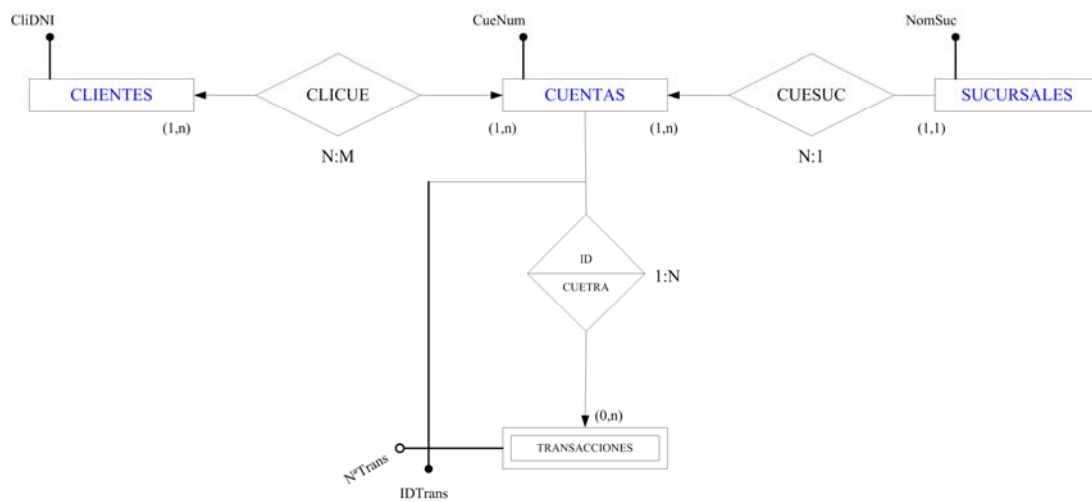
### Esquema relacional.



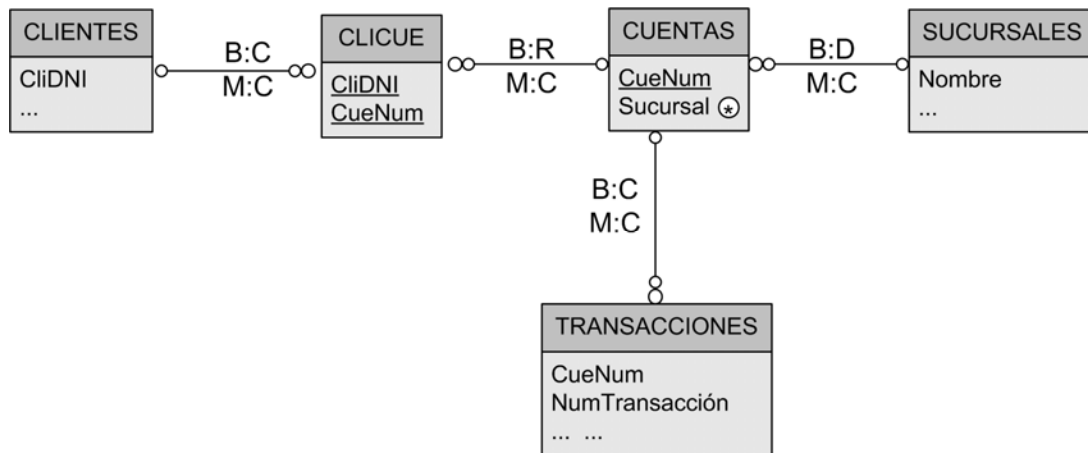
### EJERCICIO 3.

#### Esquema Entidad Relación Extendido.



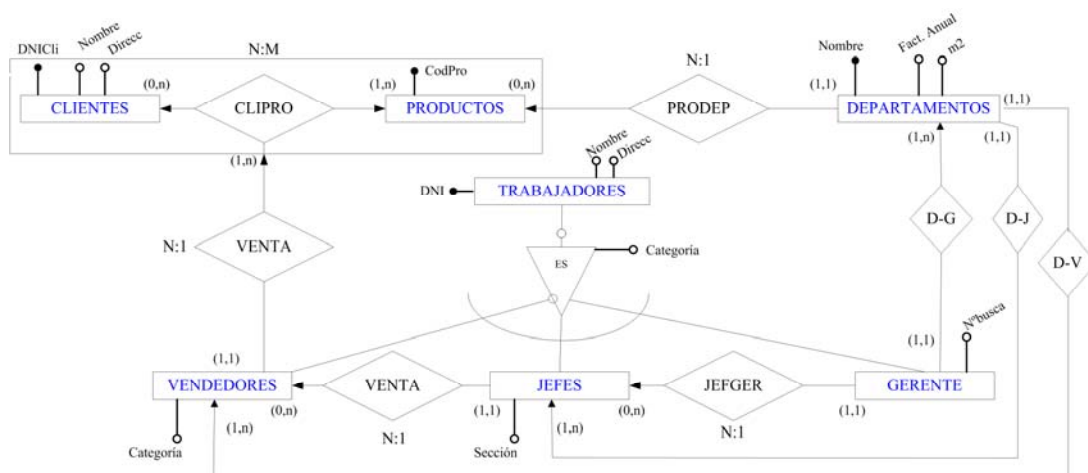
**Esquema relacional.****EJERCICIO 4.****Esquema Entidad Relación Extendido.**

### Esquema relacional.

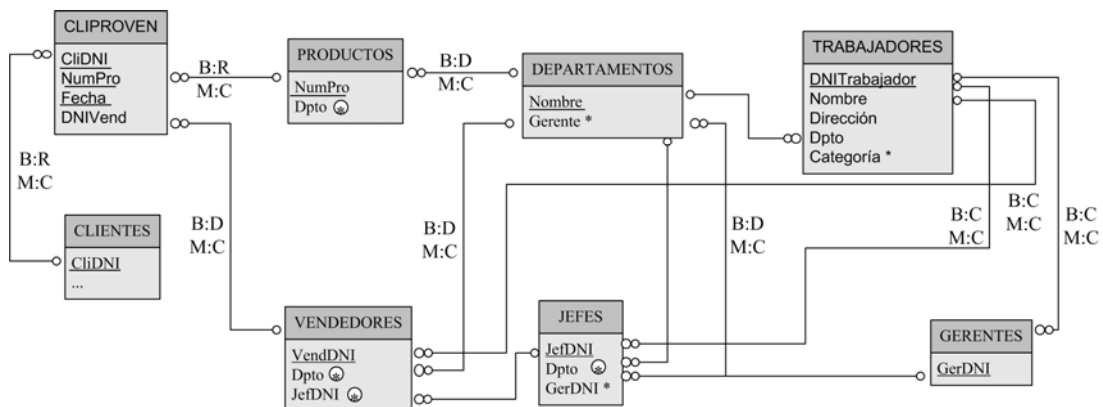


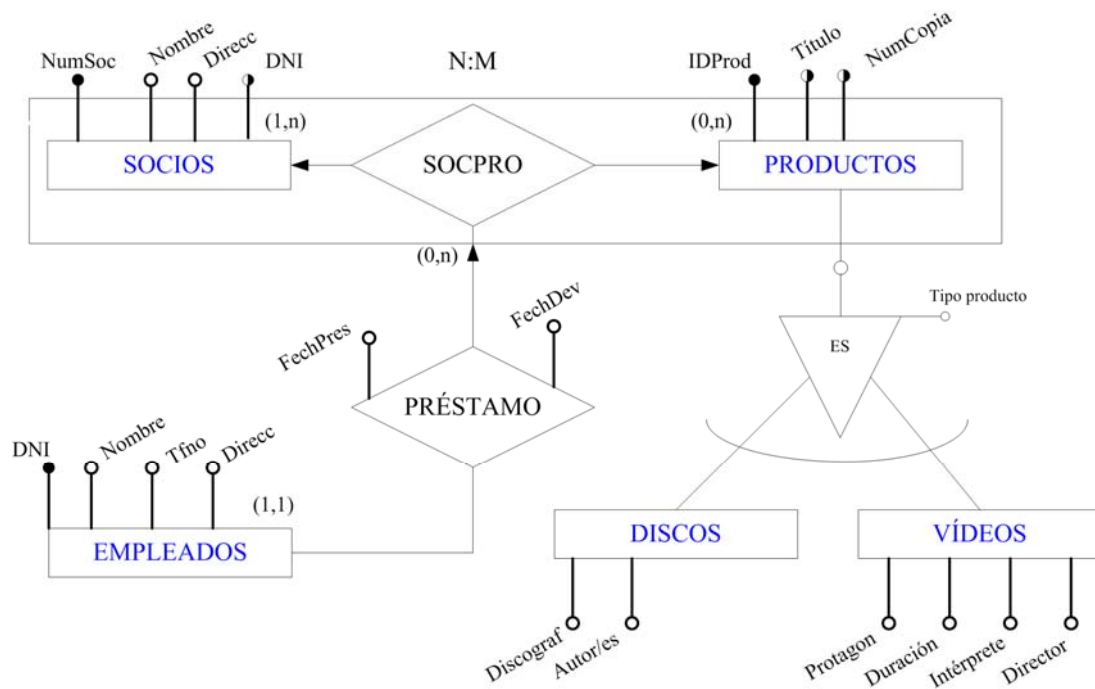
### EJERCICIO 5.

#### Esquema Entidad Relación Extendido.



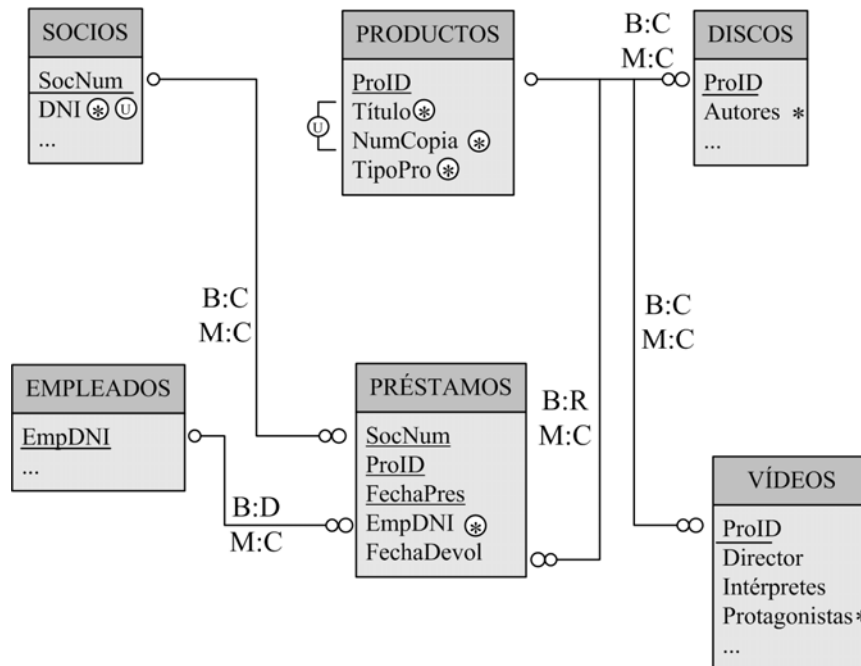
### Esquema relacional.



**EJERCICIO 6.****Esquema Entidad Relación Extendido.**

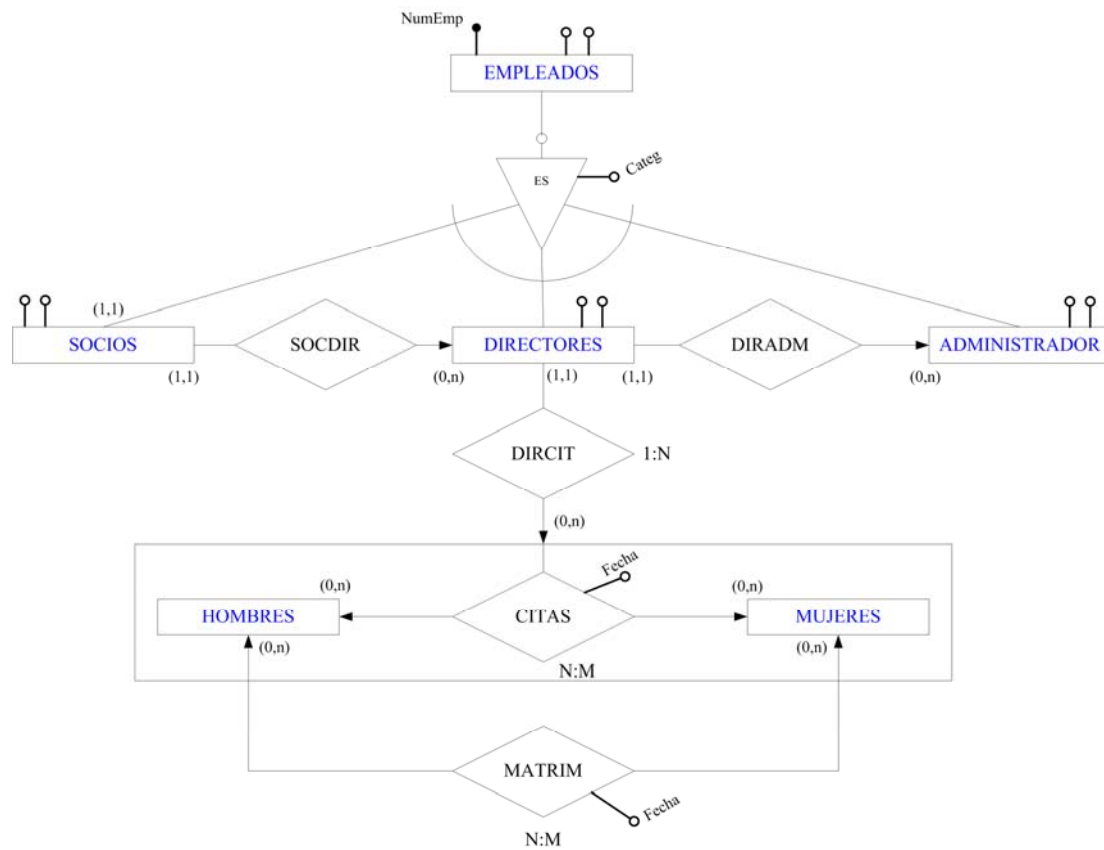


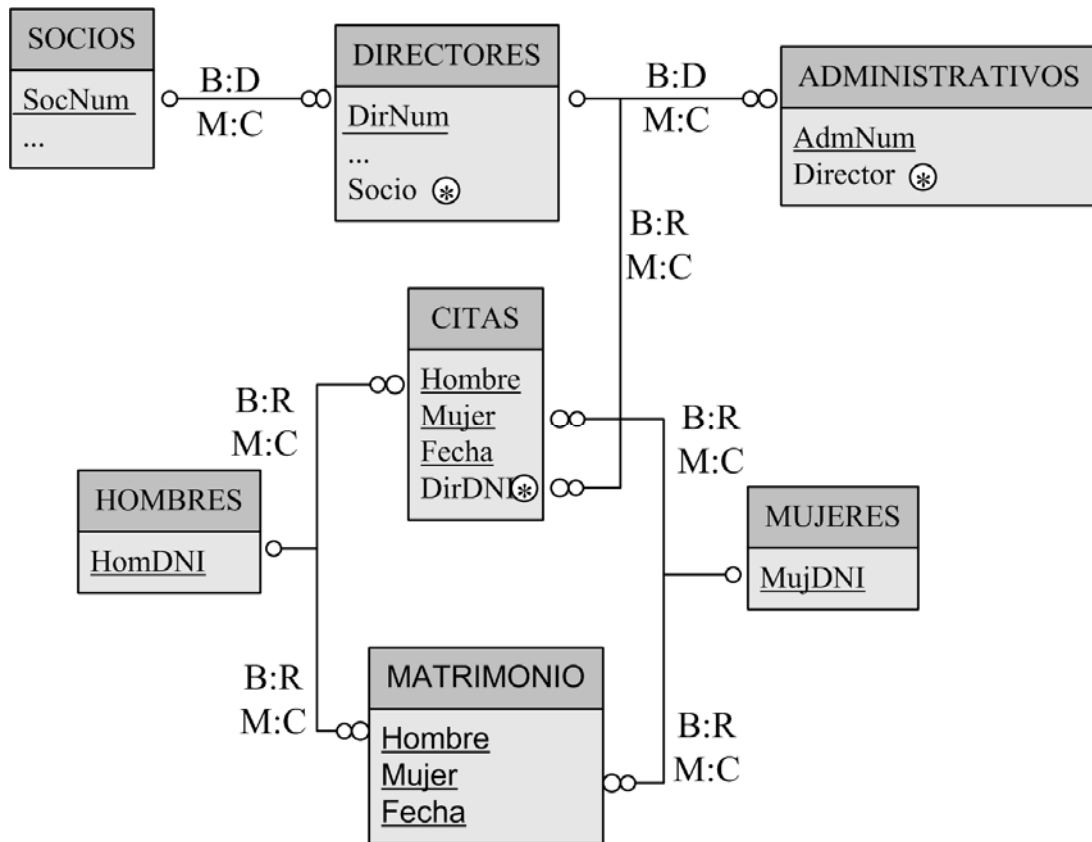
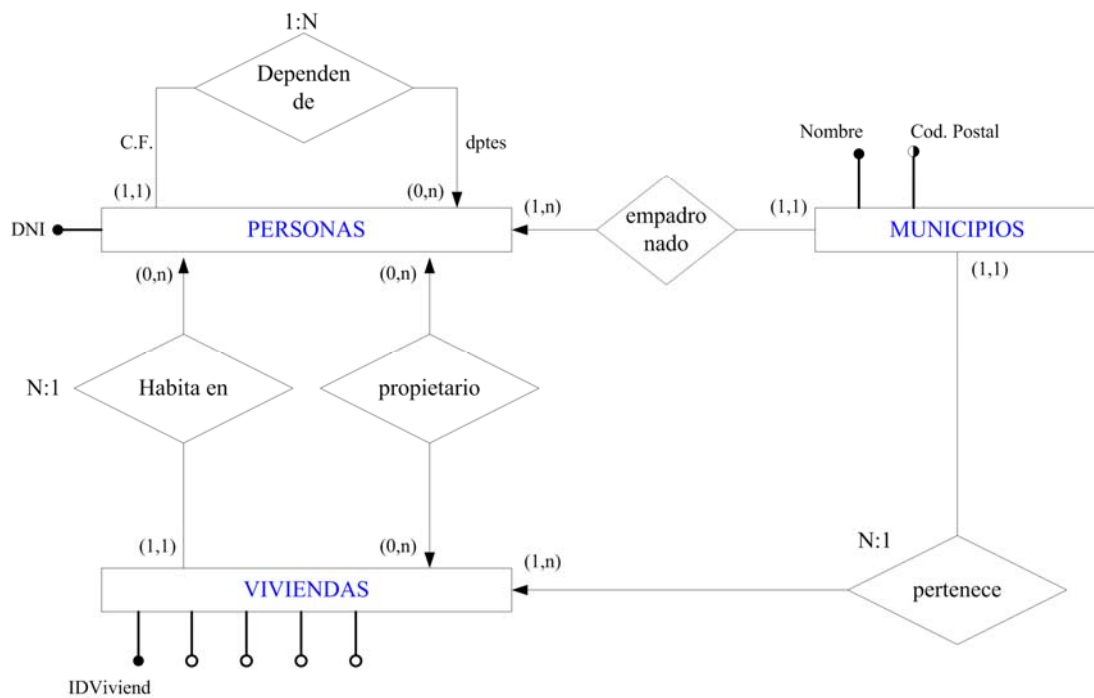
### Esquema relacional.



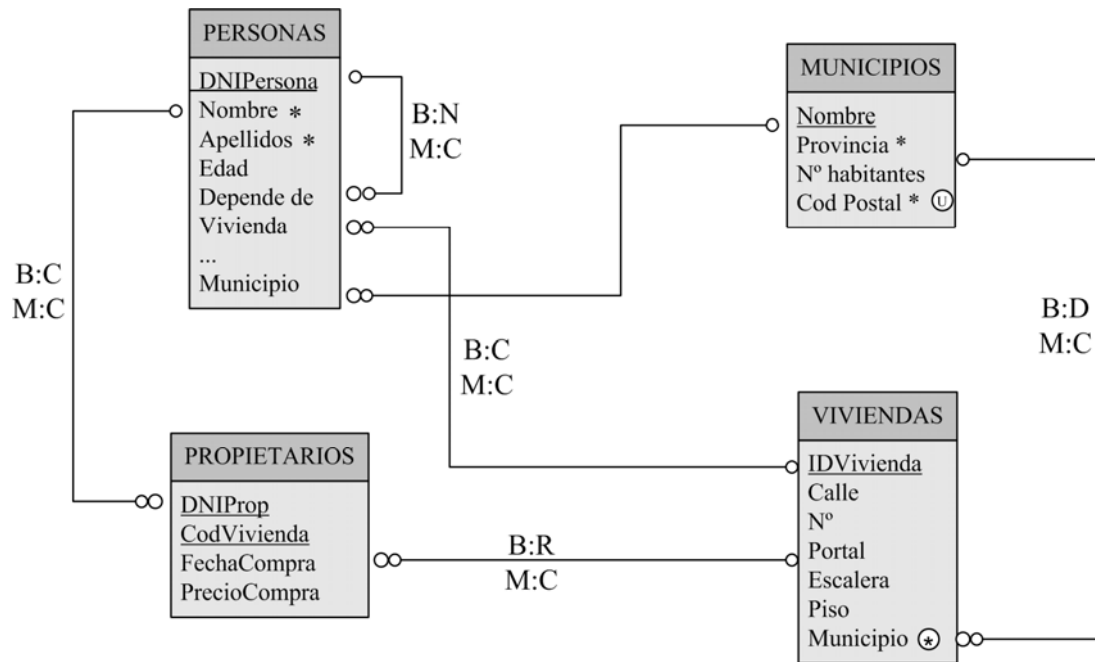
### EJERCICIO 7.

#### Esquema Entidad Relación Extendido.

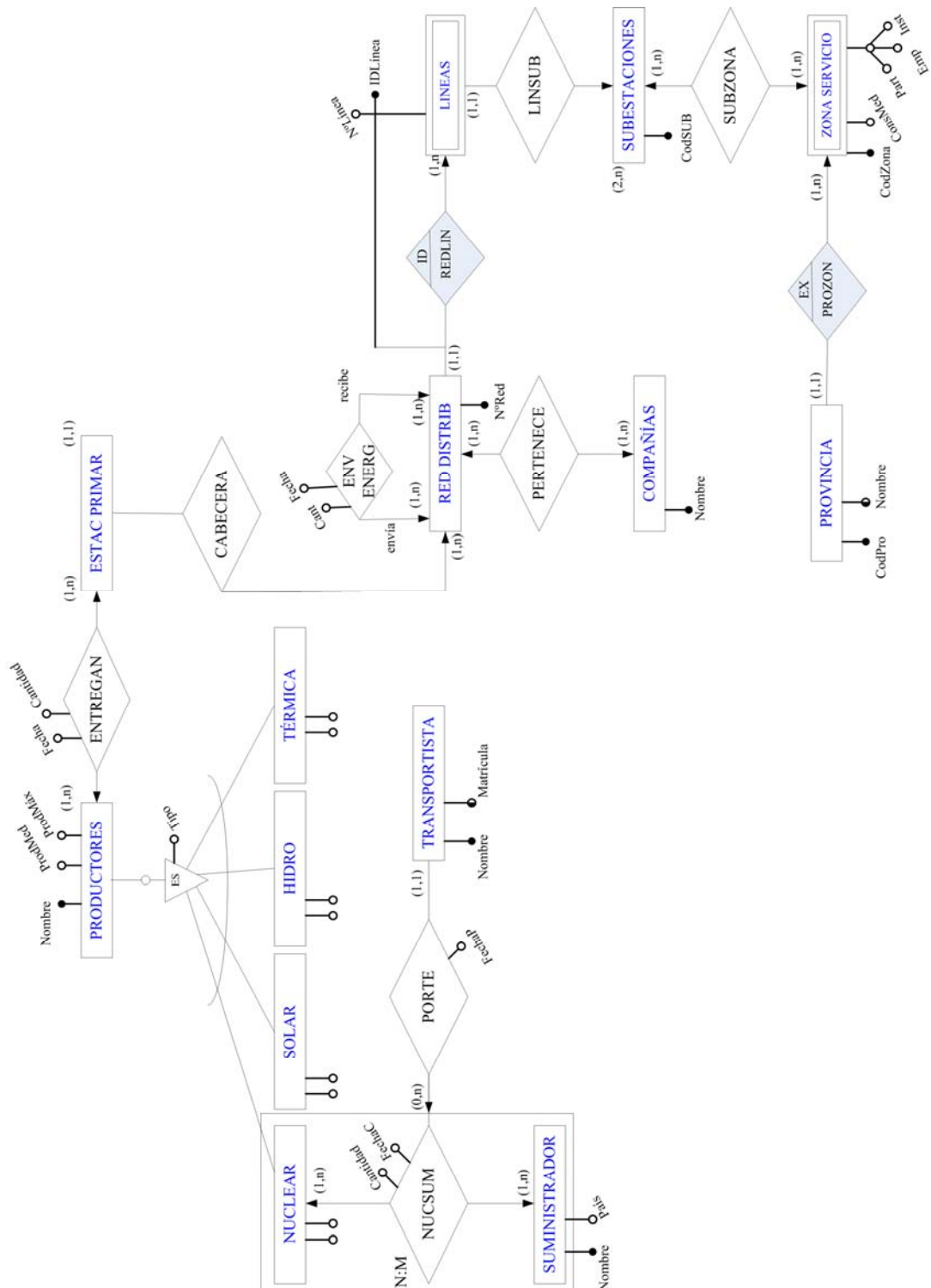


**Esquema relacional.****EJERCICIO 8.****Esquema Entidad Relación Extendido.**

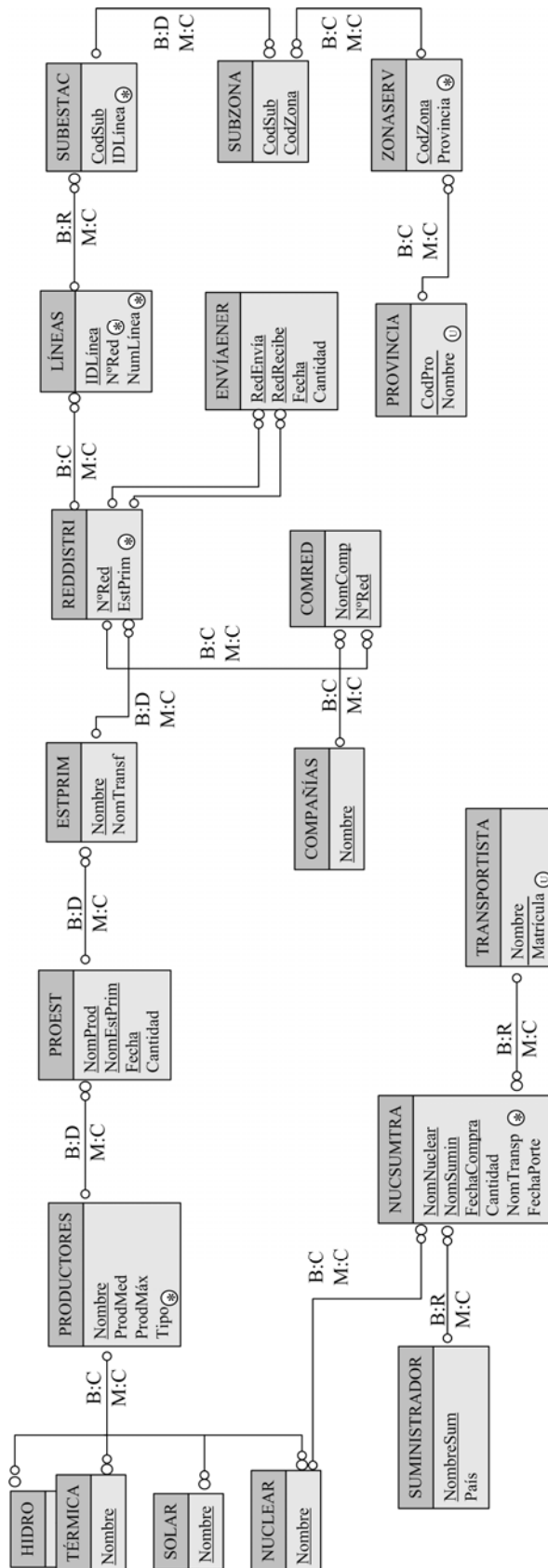
### Esquema relacional.

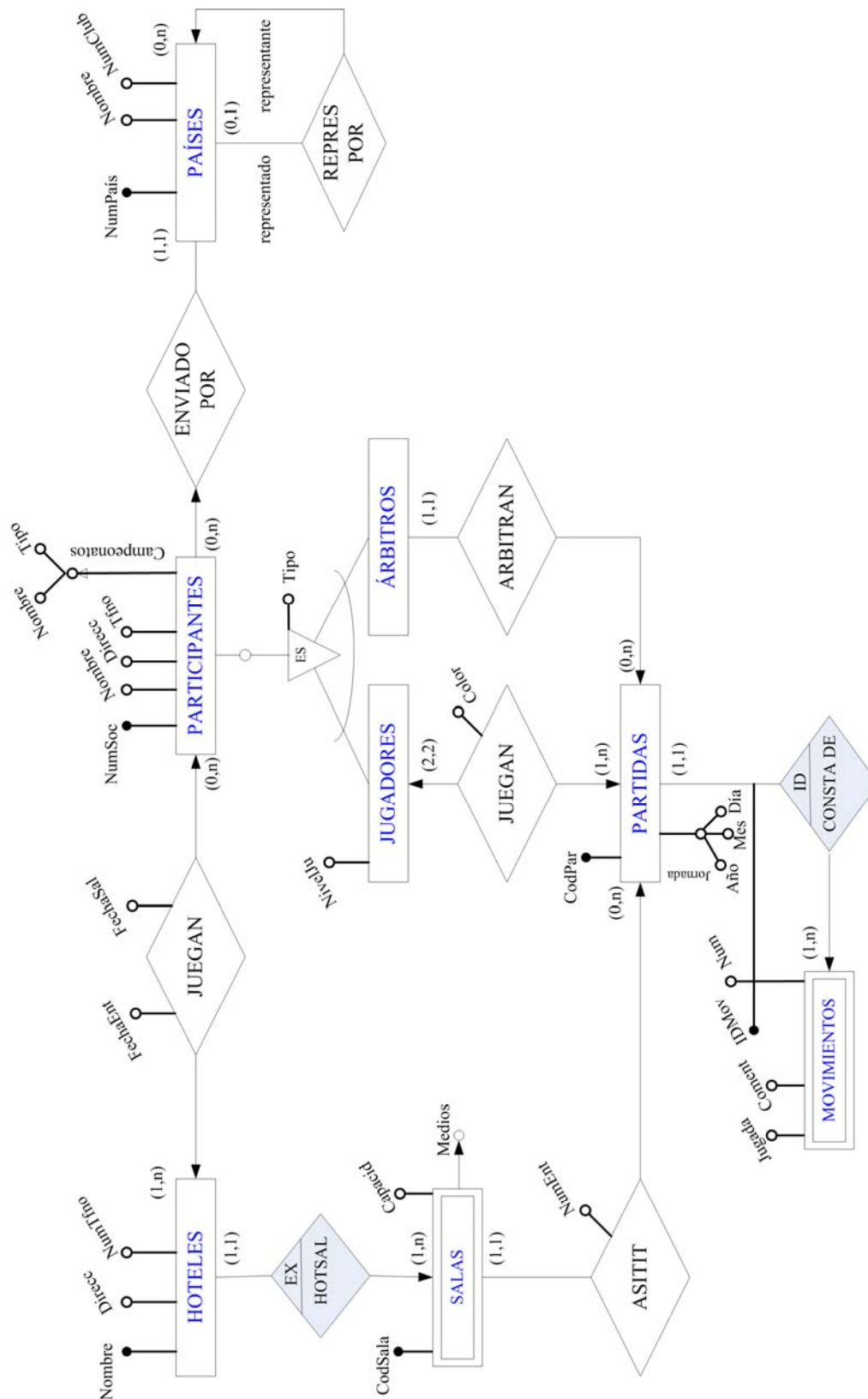


En la tabla SERPROPIETARIO, la clave primaria está formada por DNI e IDVivienda porque una persona puede poseer varias viviendas. En la tabla VIVIENDAS, Municipio es no nulo porque todas las viviendas pertenecen a un municipio.

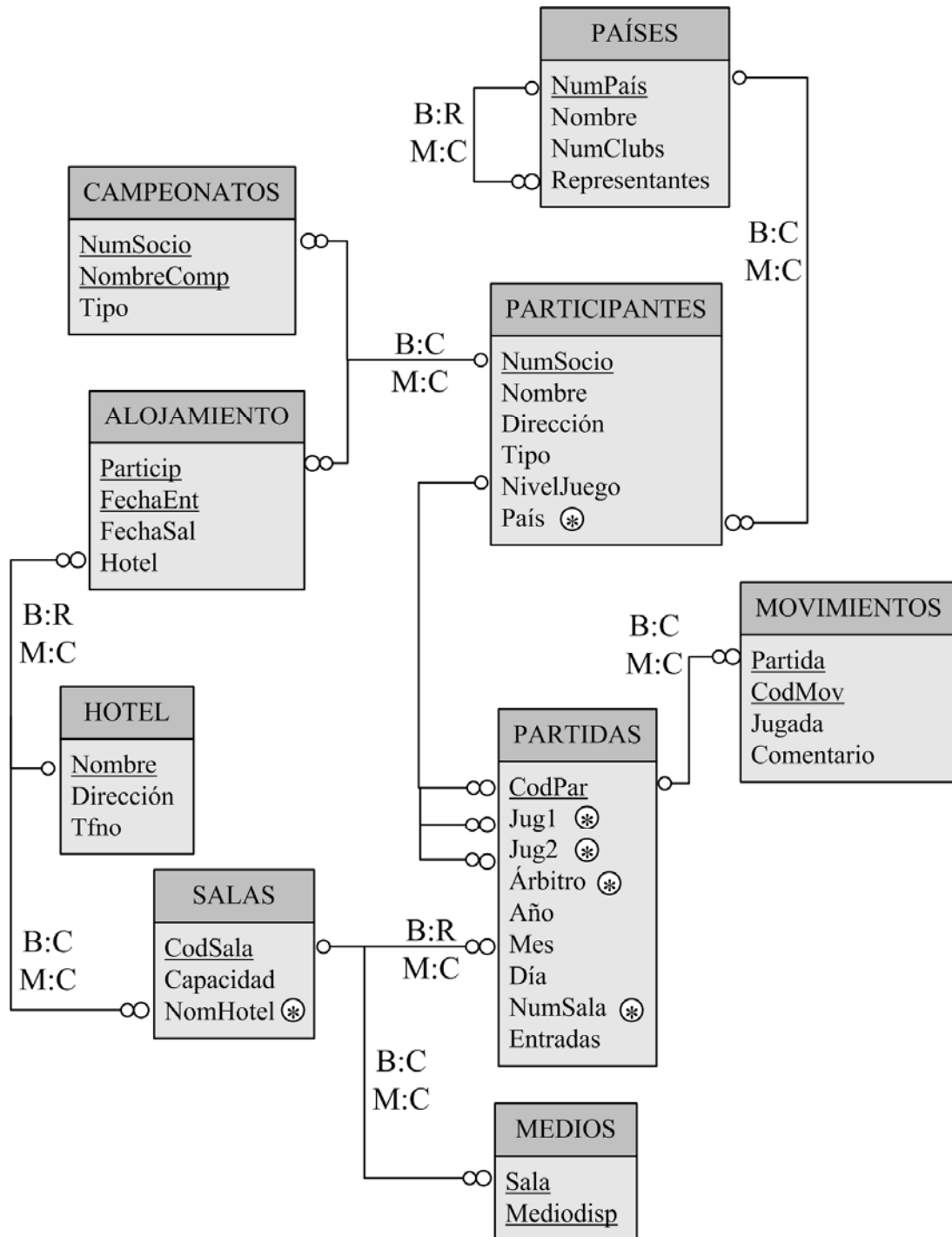
**EJERCICIO 9.****Esquema Entidad Relación Extendido.**

### Esquema relacional.



**EJERCICIO 10.****Esquema Entidad Relación Extendido.**

**Esquema relacional.**



Existe una tabla para campeonatos porque es un atributo multivaluado de participantes, mientras que jornada no tiene tabla porque es un atributo compuesto para partidas.

La clave primaria en la tabla MOVIMIENTOS es, además del número de movimiento, la partida, porque los movimientos se identifican según las partidas

(existe una dependencia de identificación). No sucede lo mismo entre salas y hoteles puesto que, en este caso, la dependencia es en existencia no en identificación, por lo que basta con que hotel sea no nulo en la tabla SALAS.

En la tabla ALOJAMIENTOS, la fecha de entrada forma parte de la clave primaria porque un mismo participante puede cambiar de lugar de alojamiento.

En la tabla PARTIDAS, los atributos no nulos se deben a que para que exista una partida es necesario que se enfrenten dos jugadores, haya un árbitro y se celebre en una sala.



# TEMA 4. NORMALIZACIÓN

## 1. TERMINOLOGÍA MODELO RELACIONAL.

Se denominan bases de datos relacionales a aquellas que utilizan el modelo relacional. Están formadas por tablas (también llamadas relaciones); las filas de estas tablas se denominan tuplas o registros. En estas tablas también se encuentran componentes anteriormente explicados como son los atributos o campos y algunas restricciones como las claves primarias y las ajenas.

## 2. TABLAS PLANAS.

### 2.1. DISEÑO.

Se dice que se utiliza una tabla plana en el diseño de una base de datos cuando se emplea una única tabla para introducir todos los datos contenidos en la misma. Utilizar tablas planas suele ser sinónimo de “no conocer los principios de modelado y diseño de bases de datos”.

Para entender mejor los inconvenientes que conlleva el uso de estas tablas se puede poner el siguiente ejemplo: se quiere construir una base de datos para una empresa constructora, en la cual se necesita almacenar los datos de los empleados contratados y de los edificios que la constructora está realizando.

La tabla plana que se obtendría para este caso sería la siguiente:

| IDTrabajador | Nombre | TipoOficio   | IDSuperior | IDEdificio |
|--------------|--------|--------------|------------|------------|
| 5            | Pepe   | Electricista | 9          | 310        |
| 5            | Pepe   | Electricista | 9          | 350        |
| 2            | Juan   | Fontanero    | Null       | 310        |
| 2            | Juan   | Fontanero    | Null       | 320        |
| 2            | Juan   | Fontanero    | Null       | 340        |
| 2            | Juan   | Fontanero    | Null       | 305        |
| 9            | Ana    | Electricista | Null       | 335        |

## 2.2. PROBLEMAS.

Los problemas con los que se aparecen al usar este diseño son los siguientes:

1. Problemas de redundancia, con lo que, para almacenar los datos, se necesita más espacio en el disco.
2. Problemas de integridad referencial. Este problema puede provocar las siguientes anomalías:
  - Actualización: cuando se quiera modificar un dato se tiene que modificar en todas sus filas para que no exista inconsistencia.
  - Borrado: en operaciones de borrado se pueden perder datos necesarios.
  - Inserción: en ocasiones no se podrán insertar datos, por ejemplo, porque la clave primaria no lo permita.

La solución para este diseño sería dividir la anterior tabla en al menos dos tablas, aunque se podría trabajar con tres tablas: una para los empleados, otra para los edificios y, por último, una que haga referencia a las relaciones empleado-edificio existentes en la empresa.

| Trabajadores |
|--------------|
| ...          |

| Asignaciones |
|--------------|
| ...          |

| Edificios |
|-----------|
| ...       |

## 3. FORMAS NORMALES.

Se utilizan para garantizar que la base de datos se encuentra en un estado estándar, normalizado.

### 3.1. PRIMERA FORMA NORMAL.

Una relación está en primera forma normal (PFN) si todos los valores de los atributos son atómicos, es decir, los atributos no toman valores que puedan definirse como conjuntos.

Ejemplo:

| IDTrabajador | Nombre | TipoOficio   | IDSuperior | IDEdificio           |
|--------------|--------|--------------|------------|----------------------|
| 5            | Pepe   | Electricista | 9          | {310, 350}           |
| 2            | Juan   | Fontanero    | Null       | {310, 320, 340, 305} |
| 9            | Ana    | Electricista | Null       | {335}                |

La tabla anterior no está en PFN dado que los atributos de IDEdificio no son atómicos.

Si se quisieran realizar búsquedas en la tabla anterior según el valor de IDEdificio se tendría que realizar una búsqueda textual en el campo IDEdificio. Estas búsquedas textuales dentro de un campo deben evitarse.

### 3.2. SEGUNDA FORMA NORMAL.

Una relación está en segunda forma normal (SFN) si cumple las siguientes condiciones:

- La relación está en primera forma normal.
- Los atributos no clave no dependen funcionalmente de parte de la clave, es decir, deben depender de todos los componentes de la clave.

Ejemplo:

| IDTrabajador | IDEdificio | Fecha inicio | Nombre |
|--------------|------------|--------------|--------|
| 5            | 310        | 10/5/99      | Pepe   |
| 2            | 310        | 20/3/99      | Juan   |
| 5            | 350        | 5/6/00       | Pepe   |
| 2            | 320        | 8/8/99       | Juan   |
| 2            | 340        | 10/2/00      | Juan   |

La clave primaria está formada por los atributos IDTrabajador e IDEdificio. No está en segunda forma normal porque el campo Nombre se conoce sólo a partir de IDTrabajador, independientemente del valor que tenga IDEdificio.

Los problemas existentes en el anterior ejemplo son los siguientes: se repiten datos y existen anomalías tanto a la hora de actualizar como a la de insertar (si no existen edificios no se podría dar de alta a un trabajador). Una posible solución sería dividir la anterior tabla en dos tablas distintas: una tabla para los trabajadores y otra para los edificios.

### 3.3. TERCERA FORMA NORMAL.

Una relación está en tercera forma normal (TFN) si cumple las siguientes condiciones:

- La relación está en segunda forma normal.
- Los valores de los atributos no clave no dependen de los valores de otros atributos no clave.

Ejemplo:

| IDTrabajador | TipoOficio   | Prima |
|--------------|--------------|-------|
| 5            | Electricista | 3,5   |
| 2            | Fontanero    | 3     |
| 9            | Electricista | 3,5   |

La clave primaria es IDTrabajador. La prima depende del campo TipoOficio, no de la clave primaria. Por tanto, no está en tercera forma normal. Debido a esto, surgen problemas:

- de espacio;
- anomalías:
  - de inserción;
  - de borrado: si desaparece un tipo de oficio, se borra la información de su prima asociada;
  - de actualización: si se cambia la prima de un oficio, se debe cambiar en todas las filas donde aparezca.

### 3.4. CUARTA FORMA NORMAL.

Una relación está en cuarta forma normal (CFN) cuando cumple las siguientes condiciones:

- i. La relación está en tercera forma normal.
- ii. Ninguno de los atributos es multivalorado.

### 3.5. QUINTA FORMA NORMAL.

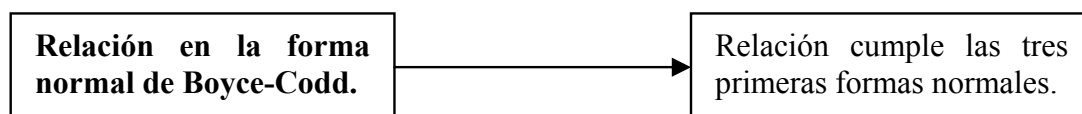
Para que una relación esté en quinta forma normal se debe evitar que existan dependencias de unión.

### 3.6. FORMA NORMAL DE BOYCE-CODD.

Una relación está en forma normal de Boyce-Codd si cada determinante de una relación es una clave primaria de dicha relación.

**Determinante:** en una dependencia funcional  $x \rightarrow y$  el determinante será el elemento denotado como  $x$ , es decir, es el atributo que determina el valor de los otros atributos de la relación.

Respecto a la forma normal de Boyce-Codd se puede enunciar la siguiente propiedad:



Nota: de forma general se debe intentar que las relaciones que se diseñen para una base de datos estén al menos en tercera forma normal.

## **4. VENTAJAS E INCONVENIENTES.**

### **a) Ventajas.**

- Facilita la comprensión del sistema de datos: al dividirlos en grupos lógicos, se comprenden mejor los datos que participan y las relaciones que tienen.
- Se ahorra espacio en disco: al evitar la redundancia.
- Es más sencillo mantener la integridad referencial dado que no hay datos redundantes.
- Se obtienen tablas estrechas, lo que conlleva las siguientes ventajas:
  - la creación y ordenación de índices es más rápida;
  - las consultas se realizan a mayor velocidad con lo que aumentará el rendimiento;
  - las entradas/salidas de datos también se realizarán a mayor velocidad debido a la organización física de los datos. En los sistemas gestores de bases de datos, los datos se almacenan en páginas. Las filas se guardan de forma completa en cada página y si no cabe una fila completa, queda vacío ese espacio en la página. Una tabla estrecha permite que haya más filas por página y, por tanto, para una consulta hay que acceder (cargar en memoria) a menos páginas.

### **b) Desventajas.**

- Al obtener tablas estrechas, aumenta el coste de recuperación de datos dado que el número de tablas que manejamos es mucho mayor.
- Las combinaciones de consultas son costosas.

## **5. DESNORMALIZACIÓN.**

### **5.1. NECESIDAD.**

La desnormalización consiste en la introducción de datos duplicados para obtener un mayor rendimiento de la base de datos.

La desnormalización se aplica cuando la base de datos es bastante estática dado que al introducir datos repetidos también se aumenta el coste de la integridad referencial.

### **5.2. REGLAS DE LA DESNORMALIZACIÓN.**

- Tener un buen conocimiento lógico y global de la base de datos.
- Desnormalizar por partes, no toda la base de datos a la vez.
- Decidir si hace falta introducir columnas virtuales.

- Se deben estudiar algunos aspectos de la base de datos a tratar como son:
  - **la frecuencia de actualización de los datos:** las partes con una gran frecuencia de actualización no se suelen desnormalizar o se hace con mucha precaución. Esto se debe a que desnormalizar consiste en introducir filas repetidas, por tanto, para actualizar hay que cambiarlas todas; esto supone muchos costes de cálculo. Cuanto mayor sea la frecuencia de actualización de los datos habrá que tener mayor cuidado en operaciones tanto de normalización como de desnormalización;
  - **aspectos de integridad de los datos:** al desnormalizar es más costoso mantener la integridad referencial;
  - **volumen de transacciones y datos:** antes de desnormalizar se intenta optimizar las consultas;
  - **aspectos de almacenamiento en disco:** cuando hay problemas en una base de datos pueden ser debidos a varias razones:
    - que estén mal diseñados los índices: habrá que modificarlos;
    - el almacenamiento interno: antes de desnormalizar se estudia agrupar en grupos de archivos y en sistemas RAID.

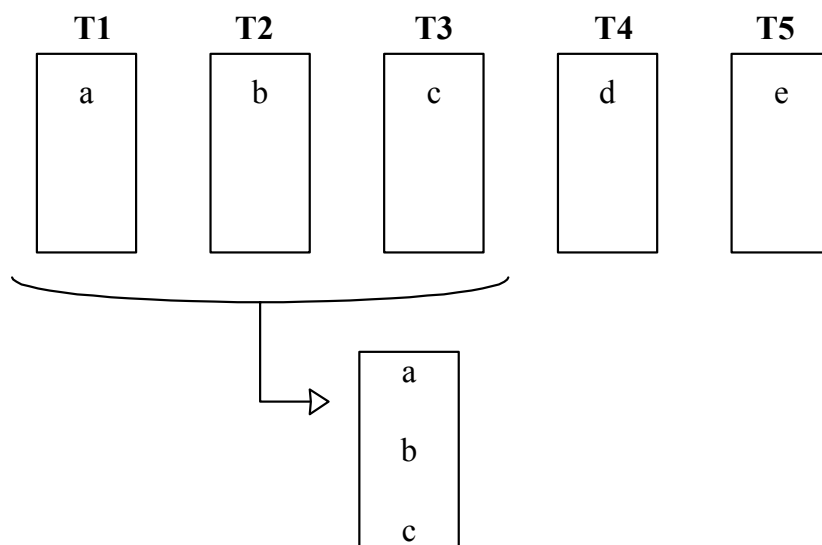
### 5.3. TÉCNICAS.

#### 5.3.1. Datos redundantes.

Este método se basa en duplicar columnas en alguna tabla. Hay que tener en cuenta que los datos han de ser estáticos y que las columnas duplicadas sean gemelas, es decir, mismo tipo de datos, misma longitud, etc.

Ejemplo:

Supongamos que se consultan con bastante frecuencia los datos a, b y c. Para que estas consultas se realicen a mayor velocidad se añadirán, a la tabla T<sub>2</sub>, las columnas a y c.



### 5.3.2. Columnas verticales.

Se definen como valores calculados aquellos que se obtienen como resultado de una agregación.

Las columnas virtuales son aquellas que contienen valores calculados. Dependiendo de la versión de SQL con la que se trabaje, el almacenamiento y tratamiento de estas columnas virtuales será distinto: en la versión 6.5. las columnas virtuales se almacenan físicamente en una tabla y se accede a ellas mediante disparadores o procedimientos almacenados, mientras que en la versión 7.0. las columnas virtuales no se guardan físicamente sino que se guarda un indicador (en la tabla del sistema) para que se sepa que la columna es calculada.

- Ventajas del uso de columnas virtuales: se puede calcular el valor de una consulta directamente, los cálculos ya se han realizado anteriormente.
- Desventajas del uso de columnas virtuales: no se pueden obtener otras columnas calculadas a partir de columnas virtuales; las columnas virtuales no pueden formar parte de agregados.

### 5.3.3. Particionamiento horizontal.

En ocasiones, se tienen tablas largas (con muchas filas) y aunque existen índices son muy poco eficientes. Para que el manejo de estas tablas sea más sencillo se parte la tabla horizontalmente en uno o en más trozos; cada una de estas subpartes obtenidas tendrá la misma estructura que la tabla original. En todo momento se debe buscar un particionamiento con sentido, que pueda responder a las posibles consultas.

Ejemplo:

Se tiene una tabla de facturación con 12 millones de registros. Para facilitar las consultas se parte la tabla en 12 subtablas asociada cada una de ellas a un mes del año.

- Ventajas del uso del particionamiento horizontal: una aplicación interesante se encuentra cuando dentro de un conjunto de datos una parte cambia mucho y otra poco. En este caso se divide la tabla principal en dos subtablas, una para los datos que sufren más modificaciones y otra para los datos que son modificados en menos ocasiones.
- Desventajas del uso del particionamiento horizontal: en muchas ocasiones las consultas se complican dado que para realizar una consulta se debe saber el nombre de la tabla y la partición en donde está.

### 5.3.4. Particionamiento vertical.

En este caso se tiene una tabla con un gran número de columnas. Teniendo en cuenta que la unidad mínima de lectura y escritura es una página, en caso de que una fila no entre en una página entera, se tendrá que pasar a la siguiente página. Por lo tanto, para acceder a los datos se deberán realizar más accesos a disco porque existen pocas filas por página.

Lo que se busca en este momento son filas estrechas; por lo tanto se divide verticalmente la tabla. Esta división debe tener cierto sentido, es decir, se pondrán

juntos aquellos datos que son más consultados y en otra tabla aquellos que sean consultados con menor frecuencia.

NOTA: tanto el particionamiento vertical como el horizontal deben realizarse antes de la producción de la base de datos. Si se hace después, se tendría que solucionar con vistas en vez de con particionamiento.



# TEMA 5. SQL

## 1. INTRODUCCIÓN.

### 1.1. ESTÁNDAR.

El SQL, lenguaje de consulta estructurado, es el lenguaje de datos estándar que utilizan las bases de datos relacionales.

### 1.2. EVOLUCIÓN.

1970: IBM, en su centro de investigación de San José (denominado actualmente Almadén), desarrolló el Sequel (lenguaje de consulta). Fue el prototipo de lo que hoy se conoce como SQL.

1986: Aparece una primera normalización ANSI, aunque no tiene éxito.

1987: IBM saca al mercado el SAA-SQL.

1989: Aparece otra normalización ANSI, y esta vez tiene más éxito, siendo Oracle el primero en utilizarlo.

1992: Sale al mercado el SQL-2; es lo que hoy día implementan todos los sistemas gestores de bases de datos, y es, asimismo, lo mínimo que se le puede pedir a un sistema de este tipo.

### 1.3. COMPONENTES.

Todo lenguaje de consulta tiene:

- a) Lenguaje de definición de datos (LDD): sirve para crear todos los objetos y datos que se utilizan normalmente (tablas, vistas, restricciones, etc.) así como las directivas de seguridad.

- b) Lenguaje de manipulación de datos (LMD): está orientado principalmente a las consultas; permite insertar datos, consultar (mediante SELECT), borrar, actualizar, etc.

#### 1.4. HERRAMIENTAS.

- Analizador de consultas: es una ventana de comandos que permite acceder a los datos de la base de datos mediante consultas con SELECT (2-1).
- “Pubs”: es la base de datos que se utilizará como ejemplo.

## 2. SELECT.

### 2.1. UTILIZACIÓN BÁSICA.

Dadas unas tablas y unas columnas, devolverá las filas que cumplan una determinada condición (2-2):

```
SELECT title
FROM titles
```

- a) Varias columnas. Selecciona todas las filas de las dos columnas indicadas de la tabla (2-3):

```
SELECT title_id, title
FROM titles
```

La sintaxis genérica es:

```
SELECT <columna1>, <columna2>
FROM <Tabla1>, <Tabla2>
```

- b) Alterando datos en el conjunto de resultados.

- Se puede utilizar como una calculadora (2-4):

```
SELECT 5*5
```

- O bien realizar operaciones sobre datos (2-5):

```
SELECT title, price, price*1.07
FROM titles
```

Las columnas que proceden de cálculos no tienen nombre. Para eso están los alias.

- c) Alias: hay tres formas de poner un alias a las columnas.

- 1) Con el ANSI (2-6):

```
SELECT au_lname Apellidos, city “Ciudad de residencia”
FROM authors
```

- 2)

```
SELECT Apellidos= au_lname, “Ciudad de residencia”= city
FROM authors
```

- 3) Éste sólo es válido para MS-SQL-7:

```
SELECT au_lname as Apellidos, city as “Ciudad de residencia”
FROM authors
```

La primera forma es la más recomendada. Los alias deben tener cierta coherencia. Si tienen signos de puntuación o blancos, deben ir entre comillas.

- d) Ver todas las columnas de una tabla. (2-7)

```
SELECT  *
FROM    jobs
```

## 2.2. INFORMACIÓN DE TABLAS, COLUMNAS Y OTROS.

- a) Sobre una base de datos (2-8): al ejecutar el siguiente procedimiento almacenado, muestra en una lista todos los objetos que hay en la base de datos: vistas, tablas de usuario, etc.

```
exec sp_help
```

- b) Información selectiva (2-9): la tabla *sysobjects* almacena información sobre los datos de todas las tablas de la base de datos. En el ejemplo, se muestran los datos de tipo usuario ("U").

```
SELECT    name
FROM      sysobjects
WHERE     type="U"
```

- c) Información sobre una tabla (2-10): se obtiene ejecutando:

```
exec sp_help authors
```

También se puede escribir de la siguiente forma (2-11):

```
SELECT    *
FROM      authors
WHERE     1=2
```

Esto devolvería sólo el nombre de las columnas porque la condición impuesta no se cumple nunca.

## 2.3. ORDER BY.

Existen tres tipos de orden:

### 1) Insensible al caso del diccionario.

Es el orden alfabético, sin importar si es mayúscula o minúscula. Se pone el primero que esté.

Sobre el conjunto {Ana, barco, Barco, ana, zapato, Zapato}, devolvería {Ana, ana, barco, Barco, zapato, Zapato}

### 2) Sensible al caso del diccionario.

Primero se ordena alfabéticamente y luego, en función de las mayúsculas y las minúsculas.

Aquí devolvería {Ana, ana, Barco, barco, Zapato, zapato }

### 3) Binario.

En este caso se ordenaría en función del código ASCII ascendente.

Devuelve {Ana, Barco, Zapato, ana, barco, zapato}

- a) Ordenar por una columna (2-12):

```
SELECT      au_lname, au_fname
FROM        authors
ORDER BY    au_lname
```

- b) Ordenar por múltiples columnas (2-13):

```
SELECT      au_lname, state
FROM        authors
ORDER BY    state, au_lname
```

- c) Ordenar descendientemente; por defecto el orden es ascendente (2-14):

```
SELECT      title_id, ytd_sales
FROM        titles
ORDER BY    ytd_sales desc
```

Los valores NULL aparecen después de todos los valores que sí se conocen.  
En orden descendente, irían los últimos.

- d) Por columnas que no aparecen en la respuesta (2-15):

```
SELECT      title_id, price*1.07 IVA
FROM        titles
ORDER BY    price desc
```

- e) Con atajos (2-16): para ordenar rápidamente se puede poner un cardinal que indica la columna.

```
SELECT      title_id, ytd_sales
FROM        titles
ORDER BY    2 desc
```

### 3. WHERE

#### 3.1. QUÉ HACE.

Limita el conjunto de datos de la respuesta a las filas que cumplan la condición (3-1):

```
SELECT      au_lname Apellidos, au_fname Nombre, state Estado
FROM        authors
WHERE       state='CA'
```

Devuelve sólo las que son de California.

#### 3.2. CONDICIONES DE IGUALDAD.

Se utilizan para encontrar un campo en concreto conocido un dato, o también para obtener un subconjunto de un conjunto (3-2):

```
SELECT      address, city, state, zip, au_lname, au_fname
FROM        authors
WHERE       lname='Hunter'
```

### 3.3. CONDICIONES DE DESIGUALDAD.

Los operadores son (3-3):

<, <=, >=, > [menor que, menor o igual, mayor o igual, mayor que]  
 <>, != [distinto]  
 !> [no mayor que]  
 !< [no menor que]

```
SELECT    title, price
FROM      titles
WHERE     price < $13
```

El símbolo \$ podría suprimirse, aunque es más eficiente ponerlo porque así se indica que el valor que le sigue es de tipo monetario, y lo trata como un *float* con punto fijo.

### 3.4. LÓGICA BOOLEANA.

a) Operadores:

AND, OR, NOT, XOR

b) Cláusulas compuestas con lógica booleana (3-4):

```
SELECT title_id, type, price
FROM   titles
WHERE  type='business' OR price<$18
```

c) Importancia del orden de las operaciones (3-5): se indica con paréntesis.

```
SELECT au_fname, au_lname, state, contract
FROM   authors
WHERE  (state<>'CA') OR (state='CA' AND contract=Ø)
```

d) Operaciones de desigualdad y fechas (3-6):

Si se quieren obtener, por ejemplo, las ventas con fecha 19 de Noviembre, no se puede restringir con {WHERE fecha\_venta='11/19/96'}, ya que el resultado sería el conjunto vacío. Hay que incluir la hora porque por defecto utiliza las 0:00 horas. La solución correcta consiste en indicarle un rango diciendo que los datos están entre una fecha y otra (3-7):

```
SELECT *
FROM   ventas
WHERE  (fecha_venta>='11/19/96') AND (fecha_venta<'11/20/96')
```

### 3.5. BETWEEN.

- *Between* es inclusivo (incluye los extremos de los intervalos)
- *Not between* es exclusivo.

```
SELECT    f_name, l_name, hire_date
FROM      employee
WHERE     hire_date BETWEEN '1/1/91' AND '1/1/92'
ORDER BY  hire_date desc
```

El optimizador de consultas lo que hace es sustituir el *Between* por (3-8):

```
SELECT    f_name, l_name, hire_date
FROM      employee
WHERE     hire_date >= '1/1/91' AND hire_date <= '1/1/92'
```

### 3.6. IN.

Sustituye a varios OR. Con estos operadores se permite acceder a datos no consecutivos (3-9).

a) Con OR:

```
SELECT    au_id, address, city, state
FROM      authors
WHERE     au_id = '172-32-1176' OR
          au_id = '238-98-7766' OR
          au_id = '486-29-1786' OR
          au_id = '648-92-1872'
```

b) Con IN:

```
SELECT    au_id, address, city, state
FROM      authors
WHERE     au_id IN ('172-32-1176', '238-98-7766',
                   '486-29-1786', '648-92-1872')
```

Las comillas dobles se usan cuando hay números que se quieren leer como un string; si no, en general, se utiliza la comilla simple.

El *NOT IN* es el contrario del *IN*, obteniéndose datos cuyo valor no está en el conjunto indicado.

### 3.7. LIKE (O USO DE COMODINES).

Un comodín sirve para encontrar datos:

| Símbolo comodín: | Sustituye a:  |
|------------------|---|
| %                | Cualquier número de caracteres                                |
| -                | Un carácter   |
| [ ]              | Un carácter en el rango o en el conjunto<br>[b-k]      [ankh] |
| [^ ]             | Un carácter no en el rango                                    |

Ejemplo 1: Todo lo que contenga la palabra “computer”: (3-10)

```
SELECT    title_id, title
FROM      titles
WHERE     title LIKE “% computer %”
```

El optimizador no puede utilizar índices para buscar “% computer %” porque empieza con el símbolo “%”.

Ejemplo 2: Se busca a alguien que se llama McBadden, pero no se sabe si es con mayúsculas o minúsculas: (3-11)

```
SELECT    au_id
FROM      authors
WHERE     lname LIKE "Mc[Bb]adden"
```

### 3.8. VALORES NULL.

Los valores *Null* no son ni ceros ni blancos. Simplemente son valores que se desconocen. Para recuperar un valor *Null* hay que poner una condición *is Null*.

```
(3-12)
SELECT    pub_name, city, state
FROM      publishers
WHERE     state is null
```

## 4. COLUMNAS Y FUNCIONES.

### 4.1. MANIPULACIÓN DE COLUMNAS.

- a) Seleccionar y usar constantes de cadena (4-1). Se utilizan cuando se quiere un listado de columnas que tienen una cadena de texto determinada.

```
SELECT    "Hola mi nombre es", fname
FROM      employee
```

- b) Todo en una sola columna (4-2).

```
SELECT    "Hola mi nombre es" + fname Presentación
FROM      employee
```

- c) Combinación de varias columnas (4-3).

```
SELECT    au_lname + ",_" + au_fname "Nombre de autores"
FROM      authors
ORDER BY  "Nombre de autores"
```

### 4.2. CONVERT ().

Es una función que se utiliza para dar formato al texto. Su sintaxis es la siguiente:

```
convert(tipo (longitud), columna)
```

donde *tipo* es el tipo de datos al que se quiere convertir, *longitud* indica la longitud máxima y *columna*, la columna que se desea convertir.

- a) Permite convertir un tipo de datos en otro. Por ejemplo, para la siguiente consulta (4-4):

```
SELECT    title_id + "coste $" + price
FROM      titles
```

Esta consulta daría un error porque se intenta sumar columnas de distinto tipo. La consulta correcta sería (4-5):

```
SELECT    title_id + "coste $" + convert(varchar(10), price)
FROM      titles
ORDER BY  price
```

Se convierte la columna *price* al tipo de datos *varchar* con longitud máxima 10. No formatea a un ancho fijo, es decir, si es menor a 10, ocupa el tamaño exacto, alineándolo a la izquierda sin dejar espacios en blanco.

- b) Tratamiento de datos (4-6). El siguiente ejemplo trunca la cadena a un tamaño máximo de 12 caracteres.

```
SELECT    convert(char(12), title) "Título corto", price
FROM      titles
```

- c) Formatear (dar formato) fechas (4-7). La función *convert* dispone de una serie de opciones para trabajar con fechas; en el ejemplo siguiente utiliza la opción 9 y la función del sistema *getdate*.

```
SELECT    convert(varchar(40), getdate( ), 9);
```

### 4.3. FUNCIONES.

Permiten hacer cálculos con los valores que se consultan, pero hay que tener en cuenta que disminuyen el rendimiento. De usar funciones, es mejor hacerlo en la instrucción *SELECT* en vez de en la instrucción *WHERE*.

- 1) Funciones de cadena:

- Upper, Lower;
- Substring (source, start, length): obtiene una parte de una cadena;
- Rtrim, Ltrim: recortan una cadena por la derecha o por la izquierda;
- Str (<source number>, [longitud de salida], [scale]): convierte un número a carácter ;
- ASCII, char;
- Space, reverse, replicate, etc.

- 2) Matemáticas:

- ABS, SIGN, SIN, COS, TAN, ASIN, ACOS, ATAN, +, -, \*, /

- 3) De fecha:

- Getdate;
- Dateadd;
- Datediff;
- Datepart;
- Datename.

- 4) Funciones del sistema:

- Datalength: longitud de una fecha;
- Username, susername: devuelve el nombre del usuario en el sistema;
- Tablas del sistema: object\_name, object\_id, col\_length.



## 5. AGREGADOS.

### 5.1. AGREGADOS.

Son funciones que devuelven un único valor en una columna correspondiente a un conjunto de filas. En un SELECT se pueden poner varios agregados.

- a) SUM: devuelve el resultado de sumar filas no nulas. (5-1)

```
SELECT    sum(price)
FROM      titles
```

- b) AVG: función que calcula el valor medio de una columna (*average*). No tiene en cuenta los valores *Null*. (5-2)

```
SELECT    avg(price)
FROM      titles
```

- c) MIN y MAX: devuelven el mínimo y el máximo, respectivamente. (5-3)

```
SELECT    min(price) "El más barato", max(price) "El más caro"
FROM      titles
```

- d) COUNT: devuelve una columna en la que aparece el número de filas que cumplen cierta condición.

- i. Las que cumplen la condición. Devuelve el número de filas que tienen un precio con valor no *Null*. (5-4)

```
SELECT    count(price)
FROM      titles
```

- ii. Todas las filas. Para ello, se utilizan comodines. (5-5)

```
SELECT    count(price) "Título con precio", count(*)
          "Totales"
FROM      titles
```

- iii. Número de filas con valores diferentes. (5-6)

```
SELECT    count(distinct price) "Títulos con distinto precio"
FROM      titles
```

### 5.2. USO DE LOS AGREGADOS.

Los tipos de datos que utilizan los agregados son:

- Count: cualquier tipo.
- Sum, avg: numéricos.
- Max, min: numéricos, carácter, fecha.

### 5.3. CON WHERE.

Restringe los valores sobre los que opera el agregado. Primero se calcula el WHERE y con el resultado se aplica el agregado. Importante: no se pueden incluir agregados dentro de las cláusulas WHERE. La siguiente instrucción devuelve el precio medio de los libros que son de tipo *business*. (5-7)

```
SELECT    avg(price)
FROM      titles
WHERE     title = 'business'
```

### 5.4. CON EXPRESIONES.

Dentro de un agregado se puede poner cualquier expresión: sumas, restas, etc. Si se quieren calcular los ingresos anuales por libro (precio \* unidades vendidas) sería: (5-8)

```
SELECT      titles_id, price*ytd_sales "Ingresos anuales por libros"
FROM        titles
```

Para calcular la suma de todos los libros (≈ingreso total por libros): (5-9)

```
SELECT      sum(price*ytd_sales) "Ingreso total"
FROM        titles
```

### 5.5. IS NULL ( , ).

Permite modificar un valor nulo por una expresión.

ISNULL(<expresión>, <valor>).

Por ejemplo, cambiar los precios con valor Null y asignarles el valor 35: (5-10)

```
SELECT      title_id, convert(char(12), title) Título, price Precio,
            isnull(prices, $35) "Precio propuesto"
FROM        titles
ORDER BY    price
```

## 6. TOTALES Y SUBTOTALES.

Para hacer informes en los que se quieren calcular totales y subtotales, y para tener en cuenta resultados o hacer agregados sólo de unas filas, se utiliza lo siguiente:

### 6.1. GROUP BY.

Permite agrupar un conjunto de filas que tengan un valor común.

- a) Precio medio de los libros por categoría. (6-1)

```
SELECT      type, avg(price)
FROM        titles
```

Esta instrucción devolvería un error; en SELECT no pueden ir juntos agregados y no agregados, salvo que los no agregados vayan luego en la instrucción GROUP BY.

- b) Agrupar las filas que tienen la misma categoría y luego obtener el precio medio. (6-2)

```
SELECT      type, avg(price) "Precio medio"
FROM        titles
GROUP BY    type
```

- c) Agrupar con varias columnas. (6-3)

```
SELECT      stor_id, ord_num, sum(qty) "Libros pedidos"
FROM        sales
GROUP BY    stor_id, ord_num
```

## 6.2. CUBE Y ROLLUP.

Estas funciones calculan subtotales de una columna.

- I. Cube: calcula todos los subtotales que encuentra.
  - a) Con una fila. Se obtiene el precio medio de todos los precios medios anteriores. (6-4)
 

```
SELECT      type, avg(price) Medio
FROM        titles
GROUP BY    type with cube
```
  - b) Con varias filas. Hace los grupos con el que se especifique primero y después con el siguiente. (6-5)
 

```
SELECT      stor_id, ord_num, sum(qty) "Libros pedidos"
FROM        sales
GROUP BY    stor_id, ord_num with cube
```
- II. Rollup: sólo agrupa por *stor\_id*. Cube agruparía primero por *stor\_id* y luego por *ord\_num*. (6-6)
 

```
SELECT      stor_id, ord_num, sum(qty) "Libros pedidos"
FROM        sales
GROUP BY    stor_id, ord_num with Rollup
```

## 6.3. COMPUTE.

Realiza cálculos de agregados anidados. La columna que aparece en COMPUTE tiene que aparecer también en SELECT.

- a) Para una columna: el ejemplo siguiente calcula cuál es el mayor de los precios medios agrupados por categoría. (6-7)
 

```
SELECT      type, avg(price)
FROM        titles
GROUP BY    type
COMPUTE     max(avg(price))
```
- b) Con agregados de agregados para varias columnas. (6-8)
 

```
SELECT      title_id, advance, price
FROM        titles
GROUP BY    title_id
COMPUTE     sum(advance), avg(price)
```
- c) COMPUTE... BY: se utiliza para calcular agregados de subtotales. Se tiene que utilizar siempre con ORDER BY para ordenar las filas de forma que el subtotal tenga sentido. (6-9)
 

```
SELECT      title_id, type, advance
FROM        titles
GROUP BY    type
COMPUTE     max(advance) BY type
```

Se pueden utilizar juntos COMPUTE y COMPUTE... BY para tener información de totales y subtotales. Por ejemplo, para obtener el valor máximo de todos los máximos parciales. (6-10)

```

SELECT      title_id, type, advance
FROM        titles
GROUP BY    type
COMPUTE     max(advance) BY type          //Subtotales
COMPUTE     max(advance)                  //Total

```

#### 6.4. HAVING.

Restringe las filas que se devuelven, pero sin restringir las filas que forman parte del cálculo. Se utiliza solamente con GROUP BY para filtrar resultados. El ejemplo siguiente devuelve las que tengan precio medio mayor que 12. (6-11)

```

SELECT      type, avg(price)
FROM        titles
GROUP BY    type
HAVING      avg(price) > $12

```

### 7. UN POCO DE TODO.

#### 7.1. TABLAS DE UNIÓN.

Cuando se quiere hacer una consulta y que en el WHERE aparezcan columnas de tablas distintas, hay que utilizar uniones. Para unir dos tablas, éstas han de tener una relación a la cual se llama la *clave común*. Una *clave común* o *join key* es la columna compartida por las dos tablas.

Ejemplo (7-1):

```

SELECT      convert(char(20), titles.title) Título, convert(varchar (10),
sales.ord_date,3), sales.qty
FROM        titles INNER JOIN sales ON titles.title_id=sales.title_id

```

##### A. Utilizando "ALIAS" de tablas. (7-2)

```

SELECT      p.pub_name, e.fname+" "+e.lname "Nombre
empleado", e.job_lvl
FROM        publishers p INNER JOIN employee e ON
p.pub_id=e.pub_id
WHERE       e.job_lvl > 200

```

##### B. Uniendo múltiples tablas (tablas auxiliares y principales). (7-3)

```

SELECT      a.au_lname, a.au_fname, convert(char(20), t.title)
Título
FROM        authors a INNER JOIN titleauthor ta ON
a.au_id=ta.au_id
INNER JOIN titles t ON ta.title_id=t.title_id
ORDER BY    Título

```

##### C. Uniendo múltiples tablas que comparten una JOIN KEY. Cuando aparecen varias tablas, si sólo se especifica un camino, el optimizador no podrá realizar su tarea. Para ello, se utilizan operadores ...ON...AND..., ya que se dan varios caminos y el optimizador elige el camino que haga más óptima la consulta. (7-4)

```

SELECT      a.au_lname, a.au_fname, sum(t.price*s.qty) "Ingresos
            totales"
FROM        authors a JOIN titleauthor ta ON a.au_id=ta.au_id
            JOIN sales s ON ta.title_id=s.title_id
            JOIN titles t ON s.title_id=t.title_id AND
            ta.title_id=t.title_id
GROUP BY    au_lname, au_fname
ORDER BY    "Ingresos totales"

```

## 7.2. SUBCONSULTAS.

Una subconsulta es una consulta normal que se encuentra anidada, utilizando paréntesis, dentro de otra. Permiten resolver, en un solo paso, problemas que de otra manera requerirían múltiples pasos.

- A. Devolviendo un único resultado. Se utiliza el resultado de una consulta sin tener que emplear una variable que almacene el resultado. (7-5)

```

SELECT      max(hire_date)
FROM        employee
➔          May 1 1994 12:00 AM

SELECT      fname, lname
FROM        employee
WHERE       hire_date="May 1 1994 12:00AM"

```

Para calcular el último empleado contratado:

```

SELECT      fname, lname
FROM        employee
WHERE       hire_date=(SELECT max (hire_date)
                        FROM employee)

```

- B. Devolviendo múltiples filas. La consulta exterior debe usar el operador IN para buscar en la lista de valores devueltos. La siguiente sentencia obtiene el título que tenga una fecha de venta entre dos fechas dadas. (7-6)

```

SELECT      title
FROM        titles
WHERE       title_id NOT IN (
                SELECT title_id
                FROM sales
                WHERE ord_date>= "1/1/93" AND ord_date <
                "1/1/94")

```

- C. Subconsultas como uniones. Se pueden usar subconsultas para resolver problemas cuando se necesita información de más de una tabla. Sin embargo, para visualizar información de más de una tabla es mejor usar *join*. (7-7)

```

SELECT      pub_name
FROM        publishers p JOIN titles t ON p.pub_id=t.pub_id
WHERE       type = "business"

```

```

SELECT      pub_name
FROM        publishers
WHERE       pub_id IN (
                                SELECT pub_id
                                FROM titles
                                WHERE type="business")

```

Con una subconsulta no se repiten filas. Si con el primer ejemplo aparece un editor varias veces, con el segundo sólo aparece una vez.

### 7.3. CREACIÓN DE TABLAS.

Para crear tablas, hay que definir algunos atributos para cada columna, como, por ejemplo: tipo de datos para la columna, longitud (cuando sea necesario), si acepta o no NULL, etc. Por defecto un campo admite valores nulos.

- A. Añadiendo una nueva tabla “CLIENTES” a la base de datos *pubs*. Hay que tener en cuenta que el tipo de datos es específico al sistema gestor de bases de datos. (7-8)

```

CREATE TABLE CLIENTES (
                                CliID int not null,
                                CNombre varchar(15) not null,
                                CApelli varchar(20) not null,
                                CDirec1 varchar(55) null,
                                CDirec2 varchar(55) null,
                                CCiudad varchar(20) null,
                                CCodPos char(9) null
                                )

```

- B. Borrado de una tabla. (7-9)

```
DROP TABLE PRUEBA
```

- C. Tablas con columna identidad (equivalente al autonumérico) y valores por defecto. (7-10)

```

CREATE TABLE INVENTARIO (
                                InvID int identity(1,1),
                                IItemID int not null,
                                ICantid tinyint null default 1)

```

- D. Tablas con restricciones (clave primaria y unicidad). La clave primaria por defecto es agrupada (el orden que van a tener físicamente las filas de la tabla va a ser el de la clave primaria); para indicar que no lo sea se utiliza nonclustered. En cambio, las restricciones de unicidad son, por defecto, no agrupadas. (7-11)

```

CREATE TABLE ITEM (
                                ItemID smallint identity(1,1) constraint PK_NC_ItemID
                                primary key nonclustered,
                                IteDescri varchar(80) not null constraint UQ_NC_IteDescri
                                unique,
                                ItePrecio smallmoney not null,
                                IteCantid smallint not null default 100)

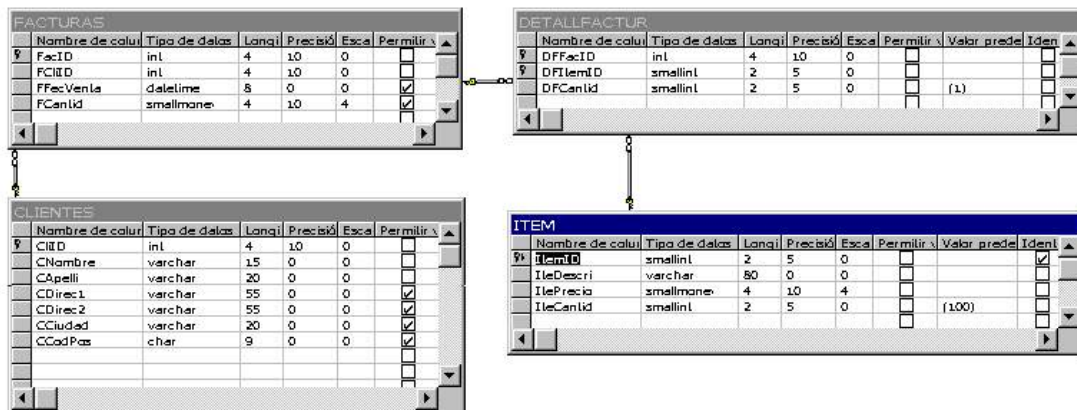
```

- E. Tabla con clave externa. El tipo identity es por defecto (1,1). En el caso de clave externa (*foreign key*) hay que señalar a qué tabla referencia y la columna o campo de la que se toman los valores, aunque este último, si no hay lugar a dudas, se puede omitir. (7-12)

```
CREATE TABLE FACTURAS (
    FacID int identity constraint PK_NC_FacID primary key
        nonclustered,
    FcliID int not null constraint FK_Facturas_FcliID foreign
        key references CLIENTES(CliID),
    FFecVenta datetime null,
    FCantid smallmoney null)
```

- F. Restricciones que afectan a restricciones. (7-13)

```
CREATE TABLE DETALLFACTOR (
    DFFacID int not null constraint
        FK_NC_detallfactor_DFFacID foreign key
        references FACTURAS,
    DFItemID smallint not null constraint
        FK_NC_detallfactor_DFItemID foreign key
        references ITEM,
    DFCantid smallint not null default 1,
    constraint PK_CL_detallfactor_FacID_ItemID
        primary key (DFFacID,DFItemID))
```



#### 7.4. INSERTANDO Y MODIFICANDO DATOS: INSERT, UPDATE, DELETE.

- A. Añadiendo un nuevo cliente. La instrucción INSERT permite introducir filas nuevas en una tabla. La columna identidad y las que tienen valores por defecto, si se quiere el valor por defecto, no se deben incluir en la instrucción. El resto de campos se especifican si se tienen datos que insertar. El siguiente ejemplo devuelve el último valor añadido en un campo identity (almacenado en la variable @@identity) para así comprobar que se insertó. (7-14)

```

INSERT CLIENTES (CNombre, CApelli, CDirec1, CCiudad,
                CCodPos)
VALUES ("José", "Fernández", "Ordoño II, 3, 1º D", "León",
        "24001")
SELECT @@identity "Ultimo valor de identidad"

```

Cómo insertar cada tipo de dato:

- ◆ Con comillas: carácter y fecha;
- ◆ Sin comillas: resto (binarios, bit y numéricos).

- B. Cambiando los valores de una fila. En el ejemplo, se actualizan los campos de la fila que cumple la condición impuesta. (7-15)

```

UPDATE CLIENTES
SET      CNombre="Ana", CApelli="Rodríguez",
        CDirec1="Independencia 2, 3º Izda.", CCiudad="León",
        CCodPos="24001"
WHERE    CliID=2
SELECT   @@identity "Ultimo valor de identidad"

```

(7-16)

```

SELECT   *
FROM     CLIENTES

```

- C. Añadiendo una nueva factura. (7-17)

```

INSERT FACTURAS (FCliID, FFecVenta, FCantid)
VALUES (1,"20/12/2000",$100.5)
SELECT @@identity "Valor de FacID"

```

- D. Creando un detalle de factura. Las inserciones deben cumplir las restricciones de integridad. El ejemplo siguiente no dejaría insertar una nueva fila ya que el campo *Item* no tiene valor. Habría que ejecutar primero el ejemplo 7-19.

(7-18)

```

INSERT DETALLFACTUR (DFFacID, DFItemID,DFCantid)
VALUES (1,1,5)

```

En la siguiente sentencia, tanto en la primera como en la tercera inserción, no se le indica un valor para el campo *IteCantid*, con lo que introduce el valor por defecto que se especificó al crear la tabla ITEM.

(7-19)

```

INSERT ITEM (IteDescri,ItePrecio)
VALUES ("Chisme azul de propósito general",$2)
INSERT ITEM (IteDescri,ItePrecio,IteCantid)
VALUES ("Chisme lila económico",$0.8,500)
INSERT ITEM (IteDescri,ItePrecio)
VALUES ("Chisme supervitaminado de lujo",$3.5)
SELECT   *
FROM     ITEM

```



(7-20)

```
INSERT DETALLFACTUR (DFFacID,DFItemID,DFCantid)
VALUES (1,1,5)
INSERT DETALLFACTUR (DFFacID,DFItemID,DFCantid)
VALUES (1,2,1)
INSERT DETALLFACTUR (DFFacID,DFItemID,DFCantid)
VALUES (1,3,4)
SELECT      *
FROM        DETALLFACTUR
```

E. Borrando filas.Delete (7-21):

```
DELETE CLIENTES
WHERE CliId=3
```

Truncate table: borra el contenido de la tabla, pero la tabla existe, aunque queda vacía. (7-22):

```
TRUNCATE TABLE INVENTARIO
```

| Operación      | ¿La tabla aún existe? | ¿Existen las filas? | ¿Logged? | ¿Se resetea identity? |
|----------------|-----------------------|---------------------|----------|-----------------------|
| Truncate table | Sí                    | No                  | No       | Sí                    |
| Delete         | Sí                    | No                  | Sí       | No                    |
| Drop table     | No                    | No                  | No       | No                    |

Donde Logged indica si queda registro de la instrucción realizada.

## 8. PROCEDIMIENTOS ALMACENADOS.

### 8.1. CONCEPTO.

Son procesos por lotes que el gestor de bases de datos almacena. Se ejecutan utilizando el nombre del proceso para realizar la llamada.

Beneficios:

#### 1) Funcionalidad.

Permiten extraer información útil del servidor incluso si no se conoce la estructura de las tablas de la que se extrae (ej.: procedimientos almacenados del sistema como sp\_help, sp\_helpdb, etc.).

#### 2) Modularidad.

sp\_help tiene unas 350 líneas de código SQL y llama a otros tres procedimientos (sp\_hempconstraint -600 líneas-, sp\_helpindex -350 líneas- y sp\_objectsegment -30 líneas); unas 1300 líneas de código en total. Escribir, modificar y depurar el mismo código en la aplicación cliente es muy costoso en tiempo de desarrollo, mantenimiento, etc.

#### 3) Mejora del rendimiento en red.

Se produce una gran sobrecarga de la red si se envían 1300 líneas de SQL cada vez que un usuario quiere ejecutar este procedimiento. Algunos procedimientos están corriendo todo el día y los usan cientos de usuarios.

#### 4) Seguridad e integridad de los datos.

Las tablas del sistema no pueden cambiarse si no es utilizando un procedimiento almacenado. Para ello se utiliza `sp_rename` que, con unas 1200 líneas, comprueba que los cambios mantienen la integridad (¿es un nombre válido de columna, es el nombre de columna único en la tabla, el usuario que lo está haciendo tiene permisos para ello?).

### 8.2. CREACIÓN.

Procedimiento que devuelve el precio de un título, conocido su identificador de título (*title\_id*).

#### a) Mediante sentencias SQL. (8-1)

```
SELECT title_id, price
FROM titles
WHERE title_id='PS7777'
```

#### b) Mediante un procedimiento almacenado.

```
CREATE proc[edure] prPrecioID
AS
SELECT title_id, price
FROM titles
WHERE title_id='PS7777'
RETURN
```

#### c) Observaciones.

- El nombre tiene que ser diferente a cualquier otro objeto de la base de datos. Hay que seguir algún convenio.
- `proc` ó `procedure`
- `AS`
- `RETURN`. Opcional pero muy recomendado.
- Mensaje del sistema tras la creación: Nada (o comando realizado satisfactoriamente).

### 8.3. INFORMACIÓN.

#### 8.3.1. Sobre un procedimiento creado.

##### a) Mediante `sp_help`. (8-2)

```
sp_help prPrecioID
```

##### b) A partir de la tabla del sistema *sysobjects*.. (8-3)

```
SELECT name, id, type
FROM sysobjects
WHERE name='prPrecioID'
```

##### c) A partir de `sp_helptext`. (8-4)

```
sp_helptext prPrecioID
```

## 8.6. EJECUCIÓN.

### a) Únicamente el procedimiento. (8-5)

Tecleando su nombre:  
prPrecioID

### b) Conjuntamente con otras sentencias. (8-6)

b.1)

```
SELECT title_id, type
FROM titles
WHERE type='business'
prPrecioID
```

b.2)

```
SELECT title_id, type
FROM titles
WHERE type='business'
execute prPrecioID
{exec=execute}
```

### c) Cambiando un procedimiento ya creado.

No se puede editar y modificar, es necesario **borrarlo** y volverlo a crear.

```
(ALTER PROCEDURE, ALTER TRIGGER ->T-SQL)
drop proc prPrecioID
```

### d) Usando parámetros. (8-7)

d.1) Declaración.

Es necesario declarar los parámetros en el CREATE.

- El tipo de datos tiene que estar relacionado con el de la columna de la tabla.
- Lo más recomendable es que sea idéntico en el tipo y longitud de los datos.
- Si hay varios parámetros se separan con comas:  
(@par1 tipo1, @par2 tipo2, ...)

```
create proc prPrecioID2
(@title_id char(6))
as
SELECT title_id, price
FROM titles
WHERE title_id=@title_id
return
```

d.2) Ejecución con un parámetro.

Se pasan los parámetros en la misma línea, después del nombre del procedimiento.

```
prPrecioID2 PS3333
```

d.3) Ejecución con varios parámetros.

Se pueden pasar por **nombre** o por **posición**.

```

create proc ejemplo
(@a int, @b int, @c int)
as
SELECT col1, col2, col3
FROM   tabla1
WHERE  a=@a AND b=@b AND c=@c
return

```

d.3.1) Por posición.

exec ejemplo 21,52,73

d.3.2) Por nombre.

exec ejemplo @a=21,@b=52,@c=73

d.3.3) Mezclados.

**BIEN:** primero posición, luego nombre.

exec ejemplo 21,@c=73,@b=52

**MAL:** primero nombre, luego posición.

exec ejemplo @b=52, 21,73

#### e) Parámetros opcionales y valores por defecto.

Declaración:

Si se declara con parámetros y no se pasan se produce un error. Para hacerlo opcional se asigna un valor por defecto en la declaración. Generalmente, se suele asignar un valor NULL y luego se toman decisiones tras comprobar el valor.

```

create proc prPrecioID3
(@title_id char(6)=null)
as
if @title_id is null
begin
print "El procedimiento prPrecioID3 requiere el parámetro
      title_id"
return
end
SELECT title_id, price
FROM   titles
WHERE  title_id=@title_id
return

```

#### f) Parámetros de salida.

Para devolver datos a partir de la ejecución de un procedimiento.

f.1) Tablas de ejemplo. (8-8)

Ejemplo: Creando tablas *articulos* y *pedidos*.

```

create table ARTICULOS
(art_id int not null, precio money null)

create table PEDIDOS
(ped_id int identity, art_id int not null, cantid int not null,
preunitar money not null, preext money not null)

```

```
insert ARTICULOS (art_id, precio) values (1, $10)
insert ARTICULOS (art_id, precio) values (2, $15)
```

f.2) Declaración.

Procedimiento que dado un identificador de artículo devuelva su precio.

```
create proc prDaPrecio
(@art_id int, @precio money output)
as
SELECT  @precio=precio
FROM    ARTICULOS
WHERE   art_id=@art_id
return
```

f.3) Ejecución. (8-9)

Declarar la variable @prevalor.

```
declare @prevalor money
exec prDaPrecio 1, @prevalor output
select @prevalor 'Precio Obtenido'
```

NOTA: Al especificar *output* para un parámetro de la definición de un procedimiento, éste podrá devolver el valor actual del parámetro al programa que lo llama cuando el procedimiento termine.

## 9. DISPARADORES (TRIGGERS).

### 9.1. CONCEPTO.

Es un tipo de procedimiento almacenado especial, asociado con una acción que se realiza sobre una tabla.

- NINGÚN usuario puede ejecutarlo.
- El sistema lo ejecuta automáticamente después de que se realice algún cambio sobre una (o varias filas) de una tabla.

**Creación.** Tabla de ejemplo y trigger que muestra el número de filas modificadas.

```
(8-10)
create table t
(c1 int identity,
c2 int not null,
c3 char(4) null)
insert t (c2, c3) values (15, 'aaaa')
insert t (c2, c3) values (25, 'bbbb')
insert t (c2) values (35)

create trigger tr
on t
for insert, update
as
```

```

raiserror ('%d filas modificadas (mensaje del
          trigger)',0,1,@ @rowcount)
return
(8-11)
insert t (c2,c3) values (45,'mmmm')

```

## 9.2. INFORMACIÓN.

Sobre un disparador creado.

**a) Mediante sp\_help. (8-12)**

```
sp_help tr
```

**b) A partir de la tabla del sistema sysobjects. (8-13)**

```

SELECT name, id, type
FROM   sysobjects
WHERE  name='tr'

```

**c) A partir de sp\_helptext. (8-14)**

```
sp_helptext 'tr'
```

## 9.3. FUNCIONAMIENTO DE LOS TRIGGERS.

**a) Ejecución de un disparador.**

Cuando se ejecuta un UPDATE sobre una tabla y hay un trigger asociado, definido también para acciones de actualización, se realizan las siguientes operaciones:

```

update t
set c3='opqr'
where c2 between 20 and 25

```

**1) Begin transaction.** Se inicia una transacción de forma implícita y se marca este momento.

**2) Borrado de datos.**

- primero se quitan los datos antiguos y sus índices;
- luego se graban los datos eliminados en la tabla **deleted**.

**3) Inserción de datos nuevos.**

- primero se insertan los datos nuevos en la tabla y los índices;
- luego se graban los datos insertados en la tabla **inserted**.

**4) Ejecución del trigger.**

En este momento, cuando ya se han modificado las filas de la tabla (de claves primarias) y dentro de la transacción, se ejecuta el trigger de actualización.

**5) Commit (o rollback) transaction.**

Se quitan los bloqueos, se escriben los cambios a disco y finaliza la transacción.

## 9.4. APLICACIONES.

### a) Tablas *inserted* y *deleted*.

Las filas insertadas y borradas son "grabadas" en las tablas *inserted* y *deleted*, respectivamente.

- Sólo son accesibles desde dentro de un trigger.
- El trigger puede usar un SELECT con estas tablas como si fueran tablas normales.

#### Ejemplo:

Disparador para mostrar el contenido de las tablas *inserted* y *deleted*.

```
create trigger tr2
on t for insert, update
as
print "===== tabla inserted ====="
SELECT * FROM inserted
print "===== tabla deleted ====="
SELECT * FROM deleted
return

insert t (c2, c3) values (99, "iii")          (8-15)
update t
set c3="ddd" where c2=99
```

### b) Función UPDATE().

Permite comprobar si se ha modificado una determinada columna en la ejecución del INSERT o del UPDATE.

- Evita pérdidas de tiempo en validación de datos cuando no hay cambios.
- Sólo puede ejecutarse dentro de un trigger.

#### Ejemplo:

Comprueba si se ha modificado la columna c2 y en caso afirmativo se asegura de que el valor insertado no sea negativo.

```
create trigger tr3
on t for insert, update
as
if update(c2)
begin
    print " Comprobando c2"
    if exists (select * from inserted where c2<0)
    begin
        raiserror ('c2 debe ser positivo o cero', 16, 10)
        rollback transaction
        return
    end
end
return
```

**Comprobación del funcionamiento.**

insert t (c2,c3) values (-20, 'zzz')

**Respuesta del servidor:**

1 filas modificadas (mensaje del trigger)

Comprobando c2

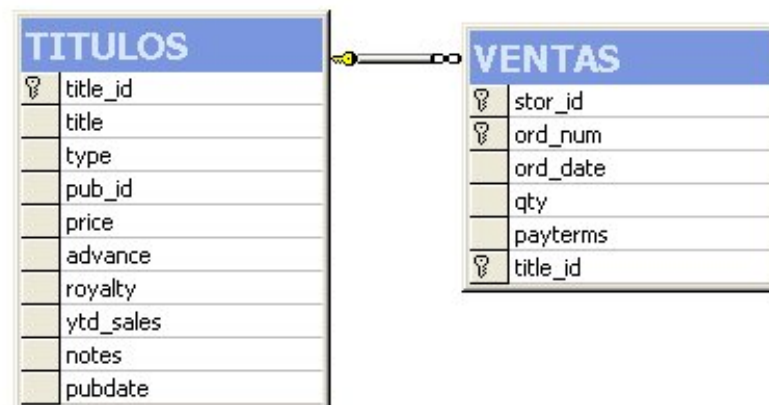
Servidor: mensaje 50000, nivel 16, estado 10, procedimiento tr2,  
línea 9

c2 debe ser positivo o cero

**c) Actualización y borrado en cascada.**

Para probarlo, se copian las tablas *titles* y *sales* de *pubs* a unas nuevas tablas de una base de datos de prueba llamadas TITULOS y VENTAS.

```
select * into TITULOS
from pubs..titles
select * into VENTAS
from pubs..sales
```

**c.1) Borrado en cascada.**

```
create trigger trTitulosD1
on TITULOS for delete
as
if @@rowcount=0 return
--Comprueba que no se han producido ventas desde hace 1 año
if exists (
    select *
    from deleted d join VENTAS v on d.title_id=v.title_id
    where datediff (dd, ord_date, getdate()) <365)
begin
    raiserror("Hay ventas recientes: borrado fallido", 16, 1)
    rollback transaction
    return
end
```

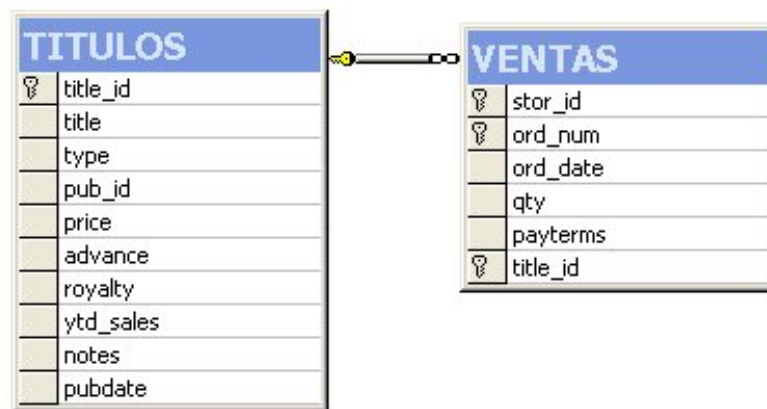


```

delete VENTAS
from deleted d
where d.title_id=VENTAS.title_id
if @@error <> 0
begin
    raiserror ("El disparador no pudo borrar las filas relacionadas
              de la tabla VENTAS",16,1)
    rollback transaction
    return
end
return

```

### c.2) Actualización en cascada.



```

create trigger trTitulosUpd
on TITULOS for update
as
if update(title_id)--si se modifica, se ejecuta
begin
    update VENTAS
    set VENTAS.title_id=inserted.title_id
    from VENTAS, inserted, deleted
    where VENTAS.title_id=deleted.title_id
    print "Tabla ventas actualizada por el disparador"
    if @@error <> 0
    begin
        raiserror ("El disparador no pudo actualizar las filas
                  relacionadas de la tabla VENTAS",16,1)
        rollback transaction
        return
    end
end
return

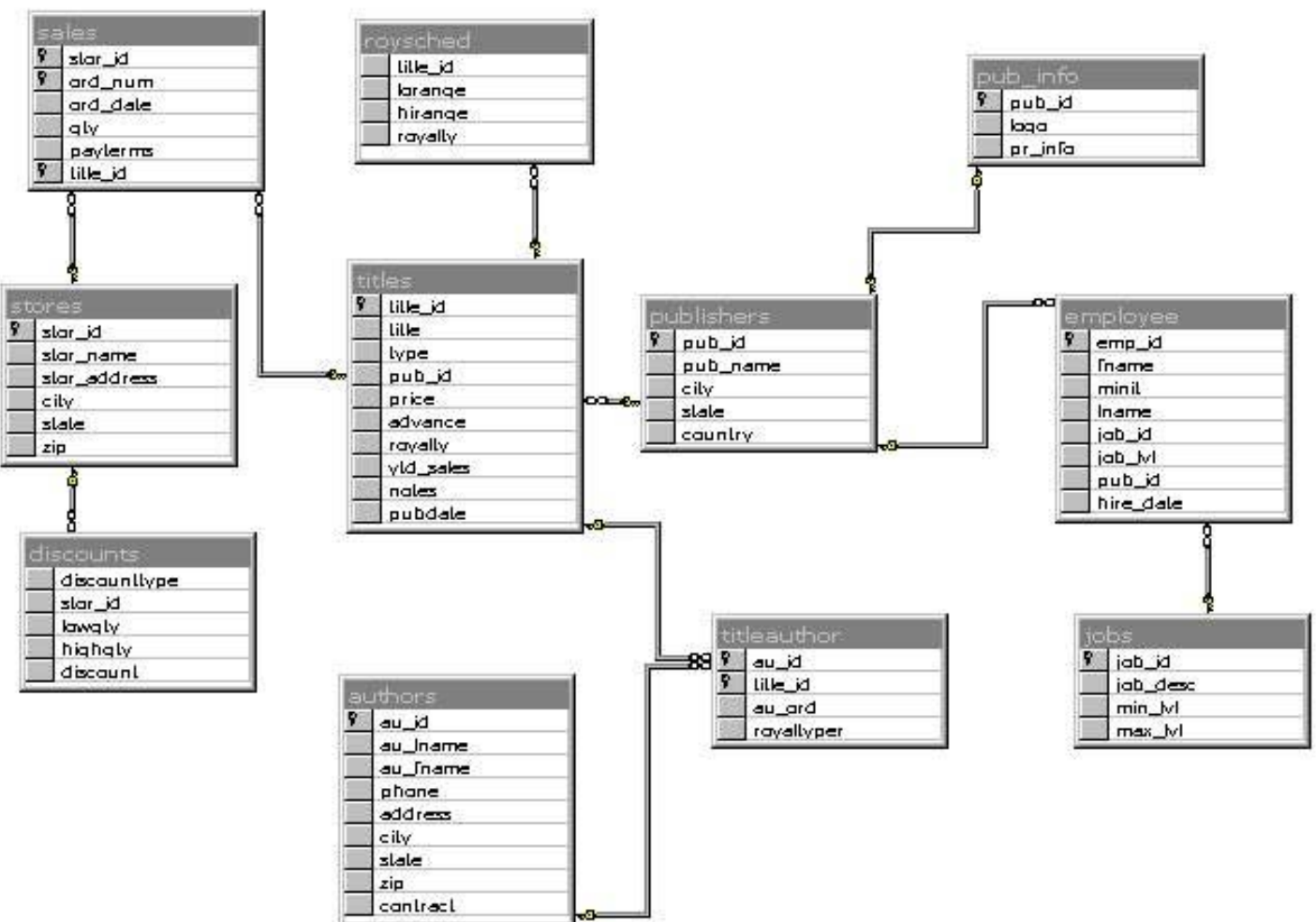
```

Ejemplo de instrucciones para comprobar que se produce la actualización:

```
insert TITULOS (title_id, title, type, pubdate)
values ('NE1111', 'Conozca su ordenador', 'hardware', '10/10/2000')
insert VENTAS (stor_id, ord_num, ord_date, qty, payterms, title_id)
values ('9001', 'K111', '1/1/2001', 3, 'Visa', 'NE1111')
update TITULOS
set title_id='NE1112'
where title_id='NE111'
```

## 10. TRANSACT SQL – EJEMPLOS.

Ejemplos realizados utilizando la base de datos “pubs”, proporcionada con Microsoft SQL Server 7.0



## 10.1. TABLAS Y CAMPOS

---

### **authors**

au\_id, au\_lname, au\_fname, phone, address, city, state, zip, contract

### **publishers**

pub\_id, pub, name, city, state, country

### **titles**

title\_id, title, type, pub\_id, price, advance, royalty, ytd\_sales, notes,  
pubdate

### **titleauthor**

au\_id, title\_id, au\_ord, royaltyper

### **stores**

stor\_id, stor\_name, stor\_address, city, state, zip

### **sales**

stor\_id, ord\_num, ord\_date, qty, payterms, title\_id

### **roysched**

title\_id, lorange, hirange, royalty

### **discounts**

discounttype, stor\_id, lowqty, highqty, discount

### **jobs**

job\_id, job\_desc, min\_lvl, max\_lvl

### **pub\_info**

pub\_id, logo, pr\_info

### **employee**

emp\_id, fname, minit, lname, job\_id, job\_lvl, pub\_id, hire\_date

Analizador de consultas de SQL Server - [Consulta - (local).pubs.sa - (sin título) - select 1+1 ...\*]

Archivo Modificar Ver Consulta Ventana Ayuda

BD: pubs

```
select 1+1

select *
from authors
```

-----

2

(1 filas afectadas)

| au_id       | au_lname     | au_fname    | phone        | address              |
|-------------|--------------|-------------|--------------|----------------------|
| 172-32-1176 | White        | Johnson     | 408 496-7223 | 10932 Bigge Rd.      |
| 213-46-8915 | Green        | Marjorie    | 415 986-7020 | 309 63rd St. #411    |
| 238-95-7766 | Carson       | Cheryl      | 415 548-7723 | 589 Darwin Ln.       |
| 267-41-2394 | O'Leary      | Michael     | 408 286-2428 | 22 Cleveland Av. #14 |
| 274-80-9391 | Straight     | Dean        | 415 834-2919 | 5420 College Av.     |
| 341-22-1782 | Smith        | Meander     | 913 843-0462 | 10 Mississippi Dr.   |
| 409-56-7008 | Bennet       | Abraham     | 415 658-9932 | 6223 Bateman St.     |
| 427-17-2319 | Dull         | Ann         | 415 836-7128 | 3410 Blonde St.      |
| 472-27-2349 | Gringlesby   | Burt        | 707 938-6445 | PO Box 792           |
| 486-29-1786 | Locksley     | Charlene    | 415 585-4620 | 18 Broadway Av.      |
| 527-72-3246 | Greene       | Morningstar | 615 297-2723 | 22 Graybar House Rd. |
| 648-92-1872 | Blotch-Halls | Reginald    | 503 745-6402 | 55 Hillsdale Bl.     |
| 672-71-3249 | Yokomoto     | Akiko       | 415 935-4228 | 3 Silver Ct.         |
| 712-45-1867 | del Castillo | Innes       | 615 996-8275 | 2286 Cram Pl. #86    |
| 722-51-5454 | DeFrance     | Michel      | 219 547-9982 | 3 Balding Pl.        |
| 724-08-9931 | Stringer     | Dirk        | 415 843-2991 | 5420 Telegraph Av.   |
| 724-80-9391 | MacFeather   | Stearns     | 415 354-7128 | 44 Upland Hts.       |

Resultados

Completado el proceso por lotes de la consulta.

Tiempo de ejecución: 0:00:00 24 filas Lin 4, Col 4

Conexiones: 1 NUM

(2-1)

## 10.2. EJEMPLOS CON SELECT

### 2-1

```
-----
2
(1 filas afectadas)
```

### 2-2

```
title
-----
But Is It User Friendly?
Computer Phobic AND Non-Phobic Individuals: Behavior
Variations
Cooking with Computers: Surreptitious Balance Sheets
Emotional Security: A New Algorithm
Fifty Years in Buckingham Palace Kitchens
Is Anger the Enemy?
Life Without Fear
Net Etiquette
Onions, Leeks, and Garlic: Cooking Secrets of the
Mediterranean
Prolonged Data Deprivation: Four Case Studies
Secrets of Silicon Valley
Silicon Valley Gastronomic Treats
Straight Talk About Computers
Sushi, Anyone?
The Busy Executive's Database Guide
The Gourmet Microwave
The Psychology of Computer Cooking
You Can Combat Computer Stress!
(18 filas afectadas)
```

### 2-3

```
title_id title
-----
PC1035 But Is It User Friendly?
PS1372 Computer Phobic AND Non-Phobic Individuals:
Behavior Variations
BU1111 Cooking with Computers: Surreptitious Balance
Sheets
PS7777 Emotional Security: A New Algorithm
TC4203 Fifty Years in Buckingham Palace Kitchens
PS2091 Is Anger the Enemy?
PS2106 Life Without Fear
.....
MC2222 Silicon Valley Gastronomic Treats
```

```

BU7832    Straight Talk About Computers
TC7777    Sushi, Anyone?
BU1032    The Busy Executive's Database Guide
MC3021    The Gourmet Microwave
MC3026    The Psychology of Computer Cooking
BU2075    You Can Combat Computer Stress!
(18 filas afectadas)

```

**2-4**

```

-----
25
(1 filas afectadas)

```

**2-5**

```

title_id price
-----
BU1032    19.9900          21.389300
BU1111    11.9500          12.786500
BU2075    2.9900           3.199300
BU7832    19.9900          21.389300
MC2222    19.9900          21.389300
MC3021    2.9900           3.199300
MC3026    NULL              NULL
PC1035    22.9500          24.556500
PC8888    20.0000          21.400000
PC9999    NULL              NULL
PS1372    21.5900          23.101300
PS2091    10.9500          11.716500
PS2106    7.0000           7.490000
PS3333    19.9900          21.389300
PS7777    7.9900           8.549300
TC3218    20.9500          22.416500
TC4203    11.9500          12.786500
TC7777    14.9900          16.039300
(18 filas afectadas)

```

**2-6**

```

Apellido          Ciudad de residencia
-----
White            Menlo Park
Green            Oakland
Carson           Berkeley
O'Leary          San Jose
Straight         Oakland
Smith            Lawrence

```

|                |                |
|----------------|----------------|
| Bennet         | Berkeley       |
| Dull           | Palo Alto      |
| Gringlesby     | Covelo         |
| Locksley       | San Francisco  |
| Greene         | Nashville      |
| Blotchet-Halls | Corvallis      |
| Yokomoto       | Walnut Creek   |
| del Castillo   | Ann Arbor      |
| DeFrance       | Gary           |
| Stringer       | Oakland        |
| MacFeather     | Oakland        |
| Karsen         | Oakland        |
| Panteley       | Rockville      |
| Hunter         | Palo Alto      |
| McBadden       | Vacaville      |
| Ringer         | Salt Lake City |
| Ringer         | Salt Lake City |

(23 filas afectadas)

**2-7**

| <b>job_id</b> | <b>job_desc</b>              | <b>min_lvl</b> | <b>max_lvl</b> |
|---------------|------------------------------|----------------|----------------|
| 1             | New Hire - Job not specified | 10             | 10             |
| 2             | Chief Executive Officer      | 200            | 250            |
| 3             | Business Operations Manager  | 175            | 225            |
| 4             | Chief Financial Officier     | 175            | 250            |
| 5             | Publisher                    | 150            | 250            |
| 6             | Managing Editor              | 140            | 225            |
| 7             | Marketing Manager            | 120            | 200            |
| 8             | Public Relations Manager     | 100            | 175            |
| 9             | Acquisitions Manager         | 75             | 175            |
| 10            | Productions Manager          | 75             | 165            |
| 11            | Operations Manager           | 75             | 150            |
| 12            | Editor                       | 25             | 100            |
| 13            | Sales Representative         | 25             | 100            |
| 14            | Designer                     | 25             | 100            |

(14 filas afectadas)

**2-8**

| <b>Name</b>         | <b>Owner</b>       | <b>Object_type</b> |
|---------------------|--------------------|--------------------|
| CHECK_CONSTRAINTS   | INFORMATION_SCHEMA | view               |
| COLUMN_DOMAIN_USAGE | INFORMATION_SCHEMA | view               |
| COLUMN_PRIVILEGES   | INFORMATION_SCHEMA | view               |



| Name                    | Owner              | Object_type   |
|-------------------------|--------------------|---------------|
| -----                   | -----              | ----- (cont.) |
| COLUMNS                 | INFORMATION_SCHEMA | view          |
| CONSTRAINT_COLUMN_USAGE | INFORMATION_SCHEMA | view          |
| CONSTRAINT_TABLE_USAGE  | INFORMATION_SCHEMA | view          |
| DOMAIN_CONSTRAINTS      | INFORMATION_SCHEMA | view          |
| DOMAINS                 | INFORMATION_SCHEMA | view          |
| KEY_COLUMN_USAGE        | INFORMATION_SCHEMA | view          |
| REFERENTIAL_CONSTRAINTS | INFORMATION_SCHEMA | view          |
| SCHEMATA                | INFORMATION_SCHEMA | view          |
| sysalternates           | dbo                | view          |
| sysconstraints          | dbo                | view          |
| syssegments             | dbo                | view          |
| TABLE_CONSTRAINTS       | INFORMATION_SCHEMA | view          |
| TABLE_PRIVILEGES        | INFORMATION_SCHEMA | view          |
| TABLES                  | INFORMATION_SCHEMA | view          |
| titleview               | dbo                | view          |
| VIEW_COLUMN_USAGE       | INFORMATION_SCHEMA | view          |
| VIEW_TABLE_USAGE        | INFORMATION_SCHEMA | view          |
| VIEWS                   | INFORMATION_SCHEMA | view          |
| Authors                 | dbo                | user table    |
| Discounts               | dbo                | user table    |
| Dtproperties            | dbo                | user table    |
| employee                | dbo                | user table    |
| jobs                    | dbo                | user table    |
| pub_info                | dbo                | user table    |
| publishers              | dbo                | user table    |
| roysched                | dbo                | user table    |
| sales                   | dbo                | user table    |
| stores                  | dbo                | user table    |
| titleauthor             | dbo                | user table    |
| titles                  | dbo                | user table    |
| employee_insupd         | dbo                | trigger       |
| sysallocations          | dbo                | system table  |
| syscolumns              | dbo                | system table  |
| syscomments             | dbo                | system table  |
| sysdepends              | dbo                | system table  |
| sysfilegroups           | dbo                | system table  |
| sysfiles                | dbo                | system table  |
| sysfiles1               | dbo                | system table  |
| sysforeignkeys          | dbo                | system table  |
| sysfulltextcatalogs     | dbo                | system table  |
| sysindexes              | dbo                | system table  |
| sysindexkeys            | dbo                | system table  |

| Name                       | Owner | Object_type      |
|----------------------------|-------|------------------|
| ----- (cont.)              |       |                  |
| sysmembers                 | dbo   | system table     |
| sysobjects                 | dbo   | system table     |
| syspermissions             | dbo   | system table     |
| sysprotects                | dbo   | system table     |
| sysreferences              | dbo   | system table     |
| systypes                   | dbo   | system table     |
| sysusers                   | dbo   | system table     |
| byroyalty                  | dbo   | stored procedure |
| dt_addtosourcecontrol      | dbo   | stored procedure |
| dt_adduserobject           | dbo   | stored procedure |
| dt_adduserobject_vcs       | dbo   | stored procedure |
| dt_checkinobject           | dbo   | stored procedure |
| dt_checkoutobject          | dbo   | stored procedure |
| dt_displayoaerror          | dbo   | stored procedure |
| dt_droppropertiesbyid      | dbo   | stored procedure |
| dt_dropuserobjectbyid      | dbo   | stored procedure |
| dt_getobjwithprop          | dbo   | stored procedure |
| dt_getpropertiesbyid       | dbo   | stored procedure |
| dt_getpropertiesbyid_vcs   | dbo   | stored procedure |
| dt_isundersourcecontrol    | dbo   | stored procedure |
| dt_removefromsourcecontrol | dbo   | stored procedure |
| dt_setpropertybyid         | dbo   | stored procedure |
| dt_validateloginparams     | dbo   | stored procedure |
| dt_vcsenabled              | dbo   | stored procedure |
| dt_verstamp006             | dbo   | stored procedure |
| dt_whocheckedout           | dbo   | stored procedure |
| reptq1                     | dbo   | stored procedure |
| reptq2                     | dbo   | stored procedure |
| reptq3                     | dbo   | stored procedure |
|                            |       |                  |
| PK__jobs__22AA2996         | dbo   | primary key cns  |
| pk_dtproperties            | dbo   | primary key cns  |
| PK_emp_id                  | dbo   | primary key cns  |
| UPK_storeid                | dbo   | primary key cns  |
| UPKCL_auidind              | dbo   | primary key cns  |
| UPKCL_pubind               | dbo   | primary key cns  |
| UPKCL_pubinfo              | dbo   | primary key cns  |
| UPKCL_sales                | dbo   | primary key cns  |
| UPKCL_taind                | dbo   | primary key cns  |
| UPKCL_titleidind           | dbo   | primary key cns  |

| Name                           | Owner | Object_type         |
|--------------------------------|-------|---------------------|
| ----- (cont.)                  |       |                     |
| FK__employee__job_id__2D27B809 | dbo   | foreign key cns     |
| FK__employee__pub_id__300424B4 | dbo   | foreign key cns     |
| FK__pub_info__pub_id__286302EC | dbo   | foreign key cns     |
| FK__roysched__title__1ED998B2  | dbo   | foreign key cns     |
| FK__sales__stor_id__1BFD2C07   | dbo   | foreign key cns     |
| FK__sales__title_id__1CF15040  | dbo   | foreign key cns     |
| FK__titleauth__au_id__164452B1 | dbo   | foreign key cns     |
| FK__titleauth__title__173876EA | dbo   | foreign key cns     |
| FK__titles__pub_id__1273C1CD   | dbo   | foreign key cns     |
|                                |       |                     |
| DF__authors__phone__09DE7BCC   | dbo   | default (maybe cns) |
| DF__dtpropert__versi__398D8EEE | dbo   | default (maybe cns) |
| DF__employee__hire_d__30F848ED | dbo   | default (maybe cns) |
| DF__employee__job_id__2C3393D0 | dbo   | default (maybe cns) |
| DF__employee__job_lv__2E1BDC42 | dbo   | default (maybe cns) |
| DF__employee__pub_id__2F10007B | dbo   | default (maybe cns) |
| DF__jobs__job_desc__239E4DCF   | dbo   | default (maybe cns) |
| DF__publisher__count__0EA330E9 | dbo   | default (maybe cns) |
| DF__titles__pubdate__1367E606  | dbo   | default (maybe cns) |
| DF__titles__type__117F9D94     | dbo   | default (maybe cns) |
|                                |       |                     |
| CK__authors__au_id__08EA5793   | dbo   | check cns           |
| CK__authors__zip__0AD2A005     | dbo   | check cns           |
| CK__jobs__max_lvl__25869641    | dbo   | check cns           |
| CK__jobs__min_lvl__24927208    | dbo   | check cns           |
| CK__publisher__pub_i__0DAF0CB0 | dbo   | check cns           |
| CK_emp_id                      | dbo   | check cns           |

| Usr_type | Storage_type | Length | Prec | Scale | Nullable | Default_name | Rule_name |
|----------|--------------|--------|------|-------|----------|--------------|-----------|
| -----    |              |        |      |       |          |              |           |
| empid    | char         | 9      | 9    | NULL  | no       | none         | none      |
| id       | varchar      | 11     | 11   | NULL  | no       | none         | none      |
| tid      | varchar      | 6      | 6    | NULL  | no       | none         | none      |

Longitud en...

- Números: nº de bytes utilizados para almacenar el número

- Cadena caracteres: número de caracteres.

Precisión: número de dígitos de un número.

Escala: número de dígitos situados a la dcha. de la coma decimal.

Ej.: 123, 45 -> precisión =5, escala =2



|       |       |
|-------|-------|
| yes   | yes   |
| yes   | yes   |
| (n/a) | (n/a) |

| Identity | Seed | Increment | Not For Replication |
|----------|------|-----------|---------------------|
|----------|------|-----------|---------------------|

|             |                 |      |           |
|-------------|-----------------|------|-----------|
| No identity | column defined. | NULL | NULL NULL |
|-------------|-----------------|------|-----------|

**RowGuidCol**

No rowguidcol column defined.

**Data\_located\_on\_filegroup**

PRIMARY

| index_name    | index_description                                    | index_keys         |
|---------------|--|--------------------|
| UPKCL_auidind | clustered, unique, primary<br>key located on PRIMARY | au_id              |
| aunmind       | nonclustered located on<br>PRIMARY                   | au_lname, au_fname |

| constraint_type         | constraint_name                 |
|-------------------------|---------------------------------|
| CHECK on column au_id   | CK__authors__au_id__08EA5793... |
| CHECK on column zip     | CK__authors__zip__0AD2A005...   |
| DEFAULT on column phone | DF__authors__phone__09DE7BCC... |
| PRIMARY KEY (clustered) | UPKCL_auidind...                |

| ... (continúa) | status_enabled | status_for_replication |
|----------------|----------------|------------------------|
|                | Enabled        | Is_For_Replication...  |
|                | Enabled        | Is_For_Replication...  |
|                | (n/a)          | (n/a) ...              |
|                | (n/a)          | (n/a) ...              |

**... (continúa) constraint\_keys**

```
(([au_id] like '[0-9][0-9][0-9]-[0-9][0-9]-[0-9][0-9]
[0-9][0-9]'))
([zip] like '[0-9][0-9][0-9][0-9][0-9]'))
('UNKNOWN')
au_id
```

**Table is referenced by**

-----  
pubs.dbo.titleauthor: FK\_\_titleauth\_\_au\_id\_\_164452B1

## 2-11

**au\_id au\_lname au\_fname phone address city statezip contract**

-----  
(0 filas afectadas)

## 2-12

| <b>au_lname</b>      | <b>au_fname</b> |
|----------------------|-----------------|
| -----                | -----           |
| Bennet               | Abraham         |
| Blotchet-Halls       | Reginald        |
| Carson               | Cheryl          |
| DeFrance             | Michel          |
| del Castillo         | Innes           |
| .....                |                 |
| Ringer               | Albert          |
| Ringer               | Anne            |
| Smith                | Meander         |
| Straight             | Dean            |
| Stringer             | Dirk            |
| White                | Johnson         |
| Yokomoto             | Akiko           |
| (23 filas afectadas) |                 |

## 2-13

| <b>au_lname</b>      | <b>state</b> |
|----------------------|--------------|
| -----                | -----        |
| Bennet               | CA           |
| Carson               | CA           |
| Dull                 | CA           |
| Green                | CA           |
| Gringlesby           | CA           |
| Hunter               | CA           |
| .....                |              |
| del Castillo         | MI           |
| Blotchet-Halls       | OR           |
| Greene               | TN           |
| Ringer               | UT           |
| Ringer               | UT           |
| (23 filas afectadas) |              |

2-14

| title_id | ytd_sales |
|----------|-----------|
| MC3021   | 22246     |
| BU2075   | 18722     |
| TC4203   | 15096     |
| PC1035   | 8780      |
| BU1032   | 4095      |
| BU7832   | 4095      |
| .....    |           |
| MC2222   | 2032      |
| PS1372   | 375       |
| TC3218   | 375       |
| PS2106   | 111       |
| MC3026   | NULL      |
| PC9999   | NULL      |

(18 filas afectadas)

2-15

| title_id | IVA      |
|----------|----------|
| PC1035   | 1.606500 |
| PS1372   | 1.511300 |
| TC3218   | 1.466500 |
| PC8888   | 1.400000 |
| BU1032   | 1.399300 |
| BU7832   | 1.399300 |
| .....    |          |
| PS7777   | .559300  |
| PS2106   | .490000  |
| BU2075   | .209300  |
| MC3021   | .209300  |
| MC3026   | NULL     |
| PC9999   | NULL     |

(18 filas afectadas)

### 10.3. EJEMPLOS CON WHERE

3-1

| Apellido  | Nombre   | Estado |
|-----------|----------|--------|
| White     | Johnson  | CA     |
| Green     | Marjorie | CA     |
| Carson    | Cheryl   | CA     |
| O'Leary   | Michael  | CA     |
| Straight  | Dean     | CA     |
| Bennet    | Abraham  | CA     |
| Dull      | Ann      | CA     |
| Gringlesb | Burt     | CA     |
| Locksley  | Charlene | CA     |
| Yokomoto  | Akiko    | CA     |
| Stringer  | Dirk     | CA     |
| MacFeathe | Stearns  | CA     |
| Karsen    | Livia    | CA     |
| Hunter    | Sheryl   | CA     |
| McBadde   | Heather  | CA     |

(15 filas afectadas)

## 3-2

| address        | city      | state | zip   | au_lname | au_fname |
|----------------|-----------|-------|-------|----------|----------|
| 3410 Blonde St | Palo Alto | CA    | 94301 | Hunter   | Sheryl   |

(1 filas afectadas)

## 3-3

| title  | price   |
|--|---------|
| Cooking with Computers: Surreptitious Balance Sheets | 11.9500 |
| You Can Combat Computer Stress!                      | 2.9900  |
| The Gourmet Microwave                                | 2.9900  |
| Is Anger the Enemy?                                  | 10.9500 |
| Life Without Fear                                    | 7.0000  |
| Emotional Security: A New Algorithm                  | 7.9900  |
| Fifty Years in Buckingham Palace Kitchens            | 11.9500 |

(7 filas afectadas)

## 3-4

| title_id | type       | price   |
|----------|------------|---------|
| BU1032   | business   | 19.9900 |
| BU1111   | business   | 11.9500 |
| BU2075   | business   | 2.9900  |
| BU7832   | business   | 19.9900 |
| PS7777   | psychology | 7.9900  |
| TC4203   | trad_cook  | 11.9500 |
| TC7777   | trad_cook  | 14.9900 |

(10 filas afectadas)

## 3-5

| au_fname    | au_lname | state | contract |
|-------------|----------|-------|----------|
| Dirk        | Stringer | CA    | 0        |
| Heather     | McBadden | CA    | 0        |
| Michel      | DeFrance | IN    | 1        |
| Meander     | Smith    | KS    | 0        |
| Morningstar | Greene   | TN    | 0        |
| Anne        | Ringer   | UT    | 1        |
| Albert      | Ringer   | UT    | 1        |

(10 filas afectadas)



## 3-6

Tabla VENTAS:

| Num_factura | FechaVenta              | Importe  |
|-------------|-------------------------|----------|
| 1           | 11/19/96 5:00:00.000 PM | \$135.16 |
| 2           | 11/19/96 7:13:41.013 PM | \$19.61  |
| 3           | 11/20/96 9:15:00.000 AM | \$344.82 |

| Num_factura         | FechaVenta | Importe |
|---------------------|------------|---------|
| (0 filas afectadas) |            |         |

## 3-7

| Num_factura | FechaVenta              | Importe  |
|-------------|-------------------------|----------|
| 1           | 11/19/96 5:00:00.000 PM | \$135.16 |
| 2           | 11/19/96 7:13:41.013 PM | \$19.61  |

## 3-8

| fname  | lname   | hire_date               |
|--------|---------|-------------------------|
| Lesley | Brown   | 1991-02-13 00:00:00.000 |
| Sven   | Ottlieb | 1991-04-05 00:00:00.000 |
| Janine | Labrune | 1991-05-26 00:00:00.000 |
| Ann    | Devon   | 1991-07-16 00:00:00.000 |
| Roland | Mendel  | 1991-09-05 00:00:00.000 |
| Aria   | Cruz    | 1991-10-26 00:00:00.000 |
| Diego  | Roel    | 1991-12-16 00:00:00.000 |

(7 filas afectadas)

## 3-9

| au_id       | address          | city          | state |
|-------------|------------------|---------------|-------|
| 172-32-1176 | 10932 Bigge Rd.  | Menlo Park    | CA    |
| 238-95-7766 | 589 Darwin Ln.   | Berkeley      | CA    |
| 486-29-1786 | 18 Broadway Av.  | San Francisco | CA    |
| 648-92-1872 | 55 Hillsdale Bl. | Corvallis     | OR    |

(4 filas afectadas)

## 3-10

| title_id | title   |
|----------|---|
| PS1372   | Computer Phobic AND Non-Phobic Individuals: Behavior Variations |
| BU1111   | Cooking with Computers: Surreptitious Balance Sheets            |
| BU7832   | Straight Talk About Computers                                   |
| MC3026   | The Psychology of Computer Cooking                              |

BU2075    You Can Combat Computer Stress!  
 (5 filas afectadas)

**3-11**

| <b>au_fname</b> | <b>au_lname</b> |
|-----------------|-----------------|
| Heather         | McBadden        |

(1 filas afectadas)

**3-12**

| <b>pub_name</b>    | <b>city</b> | <b>state</b> |
|--------------------|-------------|--------------|
| GGG&G              | München     | NULL         |
| Lucerne Publishing | Paris       | NULL         |

(2 filas afectadas)

**10.4. COLUMNAS****4-1**

| <b>fname</b>                |
|-----------------------------|
| Hola, mi nombre es Aria     |
| Hola, mi nombre es Annette  |
| Hola, mi nombre es Ann      |
| Hola, mi nombre es Philip   |
| .....                       |
| Hola, mi nombre es Sven     |
| Hola, mi nombre es Timothy  |
| Hola, mi nombre es Victoria |
| Hola, mi nombre es Yoshi    |

(43 filas afectadas)

**4-2**

| <b>Presentación</b>         |
|-----------------------------|
| Hola, mi nombre es Aria     |
| Hola, mi nombre es Annette  |
| .....                       |
| Hola, mi nombre es Victoria |
| Hola, mi nombre es Yoshi    |

(43 filas afectadas)

**4-3**

| <b>Nombres de Autores</b> |
|---------------------------|
| Bennet, Abraham           |
| Blotch-Halls, Reginald    |
| Carson, Cheryl            |
| .....                     |
| Stringer, Dirk            |
| White, Johnson            |
| Yokomoto, Akiko           |

(23 filas afectadas)

**4-4**

Servidor: mensaje 260,  
 nivel 16, estado 1,  
 línea 1

No está permitida la  
 conversión implícita del  
 tipo de datos varchar al  
 tipo money, tabla  
 'pubs.dbo.titles',  
 columna 'price'. Utilice  
 la función CONVERT para  
 ejecutar esta consulta.

**4-5**

```

-----
NULL
NULL
BU2075 coste $2.99
MC3021 coste $2.99
PS2106 coste $7.00
.....
TC3218 coste $20.95
PS1372 coste $21.59
PC1035 coste $22.95
(18 filas afectadas)

```

**4-6****Título corto price**

```

-----
The Busy Exe 19.9900
Cooking with 11.9500
You Can Comb 2.9900
Straight Tal 19.9900
.....
Onions, Leek 20.9500
Fifty Years 11.9500
Sushi, Anyon 14.9900
(18 filas afectadas)

```

**4-7**

```

select convert(varchar(40), getdate(), 0)
[ejemplo también para 3, 5, 9 y 13]

```

```

-----
Nov 28 2000 10:11PM
(1 filas afectadas)

```

```

-----
28/11/00
(1 filas afectadas)

```

```

-----
28-11-00
(1 filas afectadas)

```

```

-----
Nov 28 2000 10:11:53:317PM
(1 filas afectadas)

```

```

-----
28 Nov 2000 22:11:53:317
(1 filas afectadas)

```

**10.5. AGREGADOS****5-1**

```

-----
236.2600
(1 filas afectadas)

```

**5-2****Valor medio**

```

-----
14.7662
(1 filas afectadas)

```

Advertencia. Valor nulo  
eliminado del agregado.

**5-3****El más barato El más caro**

-----  
2.9900                  22.9500  
(1 filas afectadas)

Advertencia. Valor nulo  
eliminado del agregado.

**5-5****Titulos con precio Totales**

-----  
16                          18  
(1 filas afectadas)

Advertencia. Valor nulo  
eliminado del agregado.

**5-7****Precio medio en Business**

-----  
13.7300  
(1 filas afectadas)

**5-9****Ingresos Totales**

-----  
1045508.7200  
(1 filas afectadas)

Advertencia. Valor nulo eliminado del agregado.

Advertencia. Valor nulo  
eliminado del agregado.

**5-4**

-----  
16  
(1 filas afectadas)

Advertencia. Valor nulo  
eliminado del agregado.

**5-6****Titulos con precio diferente**

-----  
11  
(1 filas afectadas)

Advertencia. Valor nulo  
eliminado del agregado.

**5-8****Titulo Ingresos por libro**

-----  
BU1032 81859.0500  
BU1111 46318.2000  
BU2075 55978.7800  
BU7832 81859.0500  
MC2222 40619.6800  
MC3021 66515.5400  
.....  
PS7777 26654.6400  
TC3218 7856.2500  
TC4203 180397.2000  
TC7777 61384.0500  
(18 filas afectadas)

**5-10**

| <b>title_id</b> | <b>Titulo</b> | <b>Precio</b> | <b>Precio propuesto</b> |
|-----------------|---------------|---------------|-------------------------|
| MC3026          | The Psycholo  | NULL          | 35.0000                 |
| PC9999          | Net Etiquett  | NULL          | 35.0000                 |
| BU2075          | You Can Comb  | 2.9900        | 2.9900                  |
| MC3021          | The Gourmet   | 2.9900        | 2.9900                  |
| PS2106          | Life Without  | 7.0000        | 7.0000                  |
| .....           |               |               |                         |
| PC1035          | But Is It Us  | 22.9500       | 22.9500                 |

(18 filas afectadas)

**10.6. TOTALES Y SUBTOTALES****6-1**

Servidor: mensaje 8118,  
nivel 16, estado 1, línea  
1

La columna 'titles.type' de  
la lista de selección no es  
válida, porque no está  
contenida en una función de  
agregado y no hay cláusula  
GROUP BY.

**6-2**

| <b>type</b>  | <b>Precio Medio</b> |
|--------------|---------------------|
| business     | 13.7300             |
| mod_cook     | 11.4900             |
| popular_comp | 21.4750             |
| psychology   | 13.5040             |
| trad_cook    | 15.9633             |
| UNDECIDED    | NULL                |

(6 filas afectadas)

Advertencia. Valor nulo  
eliminado del agregado.

**6-3**

| <b>stor_id</b> | <b>ord_num</b> | <b>Libros pedidos</b> |
|----------------|----------------|-----------------------|
| 6380           | 6871           | 5                     |
| 6380           | 722a           | 3                     |
| 7066           | A2976          | 50                    |
| 7066           | QA7442.3       | 75                    |
| 7067           | D4482          | 10                    |
| 7067           | P2121          | 80                    |
| 7131           | N914008        | 20                    |
| 7131           | N914014        | 25                    |
| 7131           | P3087a         | 85                    |
| 7896           | QQ2299         | 15                    |
| 7896           | TQ456          | 10                    |
| 7896           | X999           | 35                    |
| 8042           | 423LL922       | 15                    |
| 8042           | 423LL930       | 10                    |
| 8042           | P723           | 25                    |

**6-4**

| <b>type</b>  | <b>Medio</b> |
|--------------|--------------|
| business     | 13.7300      |
| mod_cook     | 11.4900      |
| popular_comp | 21.4750      |
| psychology   | 13.5040      |
| trad_cook    | 15.9633      |
| UNDECIDED    | NULL         |
| NULL         | 14.7662      |

(7 filas afectadas)

Advertencia. Valor nulo  
eliminado del agregado.

8042 QA879.1 30  
(16 filas afectadas)

**6-5**

| <b>stor_id</b> | <b>ord_num</b> | <b>Libros Pedidos</b> |
|----------------|----------------|-----------------------|
| -----          | -----          | -----                 |
| 6380           | 6871           | 5                     |
| 6380           | 722a           | 3                     |
| 6380           | NULL           | 8                     |
| 7066           | A2976          | 50                    |
| 7066           | QA7442.3       | 75                    |
| 7066           | NULL           | 125                   |
| 7067           | D4482          | 10                    |
| 7067           | P2121          | 80                    |
| 7067           | NULL           | 90                    |
| 7131           | N914008        | 20                    |
| 7131           | N914014        | 25                    |
| 7131           | P3087a         | 85                    |
| 7131           | NULL           | 130                   |
| 7896           | QQ2299         | 15                    |
| 7896           | TQ456          | 10                    |
| 7896           | X999           | 35                    |
| 7896           | NULL           | 60                    |
| 8042           | 423LL922       | 15                    |
| 8042           | 423LL930       | 10                    |
| 8042           | P723           | 25                    |
| 8042           | QA879.1        | 30                    |
| 8042           | NULL           | 80                    |
| NULL           | NULL           | 493                   |
| NULL           | 423LL922       | 15                    |
| NULL           | 423LL930       | 10                    |
| .....          | .....          | .....                 |
| NULL           | TQ456          | 10                    |
| NULL           | X999           | 35                    |

(39 filas afectadas)

**6-6**

| <b>stor_id</b> | <b>ord_num</b> | <b>Libros Pedidos</b> |
|----------------|----------------|-----------------------|
| 6380           | 6871           | 5                     |
| 6380           | 722a           | 3                     |
| 6380           | NULL           | 8                     |
| 7066           | A2976          | 50                    |
| 7066           | QA7442.3       | 75                    |
| 7066           | NULL           | 125                   |
| 7067           | D4482          | 10                    |
| 7067           | P2121          | 80                    |
| 7067           | NULL           | 90                    |
| 7131           | N914008        | 20                    |
| 7131           | N914014        | 25                    |
| 7131           | P3087a         | 85                    |
| 7131           | NULL           | 130                   |
| 7896           | QQ2299         | 15                    |
| 7896           | TQ456          | 10                    |
| 7896           | X999           | 35                    |
| 7896           | NULL           | 60                    |
| 8042           | 423LL922       | 15                    |
| 8042           | 423LL930       | 10                    |
| 8042           | P723           | 25                    |
| 8042           | QA879.1        | 30                    |
| 8042           | NULL           | 80                    |
| NULL           | NULL           | 493                   |

(23 filas afectadas)

**6-7**

| <b>type</b>  |         |
|--------------|---------|
| business     | 13.7300 |
| mod_cook     | 11.4900 |
| popular_comp | 21.4750 |
| psychology   | 13.5040 |
| trad_cook    | 15.9633 |
| UNDECIDED    | NULL    |

```

max
=====
21.4750

```

(7 filas afectadas)

Advertencia. Valor nulo eliminado del agregado.

## 6-8

| title_id | advance    | price   |
|----------|------------|---------|
| PC1035   | 7000.0000  | 22.9500 |
| PS1372   | 7000.0000  | 21.5900 |
| BU1111   | 5000.0000  | 11.9500 |
| PS7777   | 4000.0000  | 7.9900  |
| .....    | .....      | .....   |
| PC9999   | NULL       | NULL    |
| TC3218   | 7000.0000  | 20.9500 |
| PS3333   | 2000.0000  | 19.9900 |
| PC8888   | 8000.0000  | 20.0000 |
| .....    | .....      | .....   |
| MC3021   | 15000.0000 | 2.9900  |
| MC3026   | NULL       | NULL    |
| BU2075   | 10125.0000 | 2.9900  |
| sum      | =====      |         |
|          | 95400.0000 |         |
|          |            | avg     |
|          |            | =====   |
|          |            | 14.7662 |

(19 filas afectadas)

Advertencia. Valor nulo eliminado del agregado.

## 6-9

| title_id | type     | advance    |
|----------|----------|------------|
| BU1032   | business | 5000.0000  |
| BU1111   | business | 5000.0000  |
| BU2075   | business | 10125.0000 |
| BU7832   | business | 5000.0000  |
|          | max      | =====      |
|          |          | 10125.0000 |
| MC2222   | mod_cook | .0000      |
| MC3021   | mod_cook | 15000.0000 |



```

max
=====
15000.0000
.....
TC3218 trad_cook 7000.0000
TC4203 trad_cook 4000.0000
TC7777 trad_cook 8000.0000
max
=====
8000.0000

MC3026 UNDECIDED NULL
max
=====

(24 filas afectadas)
Advertencia. Valor nulo eliminado del agregado.

```

**6-10**

```

title_id type advance
-----
BU1032 business 5000.0000
BU1111 business 5000.0000
BU2075 business 10125.0000
BU7832 business 5000.0000
max
=====
10125.0000

MC2222 mod_cook .0000
MC3021 mod_cook 15000.0000
max
=====
15000.0000
.....
TC7777 trad_cook 8000.0000
max
=====
8000.0000

.....
max
=====
15000.0000

```

(25 filas afectadas)

Advertencia. Valor nulo eliminado del agregado.

## 6-11

**type**

-----

business 13.7300

popular\_comp 21.4750

psychology 13.5040

trad\_cook 15.9633

(4 filas afectadas)

Advertencia. Valor nulo eliminado del agregado.

## 10.7. UN POCO DE TODO

### 7-1

**Titulo** **qty**

-----

The Busy Executive's 14/09/94 5

Is Anger the Enemy? 13/09/94 3

Secrets of Silicon V 24/05/93 50

The Gourmet Microwav 14/09/94 15

The Busy Executive's 14/09/94 10

Cooking with Compute 11/03/93 25

But Is It User Frien 22/05/93 30

(21 filas afectadas)

### 7-2

**pub\_name** **Nombre empleado** **pub\_lvl**

-----

Scootney Books Francisco Chang 227

Scootney Books Philip Cramer 215

Lucerne Publishing Carlos Hernadez 211

New Moon Books Matti Karttunen 220

Ramona Publishers Maria Pontes 246

(5 filas afectadas)

### 7-3

**au\_lname** **au\_fname**

-----

Carson Cheryl But Is It User Frien

Karsen Livia Computer Phobic AND

**MacFeather Stearns Computer Phobic AND**

**MacFeather Stearns Cooking with Compute**

O'Leary Michael Cooking with Compute

|               |               |                            |
|---------------|---------------|----------------------------|
| <b>Ringer</b> | <b>Albert</b> | <b>Is Anger the Enemy?</b> |
| <b>Ringer</b> | <b>Anne</b>   | <b>Is Anger the Enemy?</b> |
| <b>Ringer</b> | <b>Albert</b> | <b>Life Without Fear</b>   |
| DeFrance      | Michel        | The Gourmet Microwav       |
| Ringer        | Anne          | The Gourmet Microwav       |
| Green         | Marjorie      | You Can Combat Compu       |

(25 filas afectadas)

**7-4**

| <b>au_lname</b> | <b>au_fname</b> | <b>Ingresos totales</b> |
|-----------------|-----------------|-------------------------|
| -----           | -----           | -----                   |
| DeFrance        | Michel          | 119.6000                |
| Locksley        | Charlene        | 199.7500                |
| del Castillo    | Innes           | 199.9000                |
| Hunter          | Sheryl          | 1000.0000               |
| Ringer          | Anne            | 1302.2000               |
| Ringer          | Albert          | 1357.6000               |

(19 filas afectadas)

**7-5**

| <b>Fname</b> | <b>lname</b> |
|--------------|--------------|
| -----        | -----        |
| Matti        | Karttunen    |

(1 filas afectadas)

**7-6**

| <b>Title</b>  |
|---|
| -----   |
| The Busy Executive's Database Guide                             |
| The Gourmet Microwave   |
| The Psychology of Computer Cooking                              |
| Net Etiquette   |
| Is Anger the Enemy?   |
| Onions, Leeks, and Garlic: Cooking Secrets of the Mediterranean |
| Fifty Years in Buckingham Palace Kitchens                       |
| Sushi, Anyone?  |

(8 filas afectadas)

**7-7 (a)****pub\_name**

-----

Algodata Infosystems  
 Algodata Infosystems  
 New Moon Books  
 Algodata Infosystems  
 (4 filas afectadas)

**7-7 (b)****pub\_name**

-----

New Moon Books  
 Algodata Infosystems  
 (2 filas afectadas)

**7-8**

Comandos completados con  
 éxito.

**7-9**

Comandos completados con  
 éxito.

**7-10**

Comandos completados con  
 éxito.

**7-11**

Comandos completados con  
 éxito.

**7-12**

Comandos completados con  
 éxito.

**7-13**

Comandos completados con  
 éxito.

**7-14**

(1 filas afectadas)

**7-15**

(1 filas afectadas)

Ultimo valor de identidad

-----

1

(1 filas afectadas)

Ultimo valor de identidad

-----

2

(1 filas afectadas)

**7-16**

**cliID CNombre CApelli CDirec1 CDirec2**

-----

|   |      |           |                           |         |
|---|------|-----------|---------------------------|---------|
| 1 | José | Fernández | Ordoño II, 3, 1º D        | NULL... |
| 2 | Ana  | Rodríguez | Independencia 2, 3º Izda. | NULL... |

... (continúa) **Cciudad CCodPos**

-----

León 24001

León 24001

(2 filas afectadas)

**7-17**

(1 filas afectadas)

Valor de FacID

-----

1

(1 filas afectadas)

**7-18**Servidor: mensaje 547, nivel 16,  
estado 1, línea 1Instrucción INSERT en conflicto  
con la restricción COLUMN FOREIGN  
KEY 'FK\_NC\_detallfactur\_DFItemID'.  
El conflicto ha aparecido en la  
base de datos 'pubs', tabla  
'ITEM', column 'ItemID'.

Se terminó la instrucción.

**7-19**

(1 filas afectadas)

(1 filas afectadas)

(1 filas afectadas)

| ItemID | IteDescri                        | ItePrecio | IteCantid |
|--------|----------------------------------|-----------|-----------|
| 1      | Chisme azul de propósito general | 2.0000    | 100       |
| 2      | Chisme lila económico            | .8000     | 500       |
| 3      | Chisme supervitaminado de lujo   | 3.5000    | 100       |

(3 filas afectadas)

**7-20**

(1 filas afectadas)

(1 filas afectadas)

(1 filas afectadas)

| DFFacID | DFItemID | DFCantid |
|---------|----------|----------|
| 1       | 1        | 5        |
| 1       | 2        | 1        |
| 1       | 3        | 4        |

(3 filas afectadas)

**7-21**

(1 filas afectadas)

**10.8. PROCEDIMIENTOS ALMACENADOS Y DISPARADORES****8-1**

(1 filas afectadas)

| title_id | price  |
|----------|--------|
| PS7777   | 7.9900 |

(1 filas afectadas)

**8-2**

| <b>Name</b> | <b>Owner</b> | <b>Type</b>      | <b>Created_datetime</b> |
|-------------|--------------|------------------|-------------------------|
| prPrecioID  | dbo          | stored procedure | 2001-12-11 20:25:13.560 |

**8-3**

| <b>name</b> | <b>id</b>  | <b>type</b> |
|-------------|------------|-------------|
| prPrecioID  | 1285579618 | P           |

(1 filas afectadas)

**8-4**

**Text**

```

create proc prPrecioID
as
SELECT title_id, price
FROM titles
WHERE title_id='PS7777'
return

```

**8-5**

| <b>title_id</b> | <b>price</b> |
|-----------------|--------------|
| PS7777          | 7.9900       |

(1 filas afectadas)

**8-6**

Servidor: mensaje 170, nivel 15, estado 1, línea 4  
 Línea 4: sintaxis incorrecta cerca de 'prPrecioID'.

**8-7**

| <b>title_id</b> | <b>price</b> |
|-----------------|--------------|
| PS3333          | 19.9900      |

(1 filas afectadas)

**8-8**

```

select *
from ARTICULOS

```

| <b>art_id</b> | <b>precio</b> |
|---------------|---------------|
| 1             | 10.0000       |
| 2             | 15.0000       |

(2 filas afectadas)

**8-9****Precio Obtenido**

```
-----
10.0000
(1 filas afectadas)
```

**8-10**

```
select *
from t
```

| <b>c1</b> | <b>c2</b> | <b>c3</b> |
|-----------|-----------|-----------|
| 1         | 15        | aaaa      |
| 2         | 25        | bbbb      |
| 3         | 35        | NULL      |

(3 filas afectadas)

**8-11**

1 filas modificadas (mensaje del trigger)  
(1 filas afectadas)

**8-12**

| <b>Name</b> | <b>Owner</b> | <b>Type</b> | <b>Created_datetime</b> |
|-------------|--------------|-------------|-------------------------|
| tr          | dbo          | trigger     | 2001-12-12 14:41:56.733 |

**8-13**

| <b>name</b> | <b>id</b>  | <b>type</b> |
|-------------|------------|-------------|
| tr          | 1653580929 | TR          |

(1 filas afectadas)

**8-14**

```
Text
create trigger tr
on t
for insert, update
as
raiserror ('%d filas modificadas (mensaje del trigger)',0,1,@@rowcount)
return
```

**8-15**

1 filas modificadas (mensaje del trigger)  
Comprobando c2  
Servidor: mensaje 50000, nivel 16, estado 10, procedimiento tr2, línea 9  
c2 debe ser positivo o cero

## 11. TRANSACT SQL. EJERCICIOS PROPUESTOS.

- 1) Obtener un listado que contenga todos los pedidos realizados, diferenciados mediante su número de pedido, con su fecha y el identificador del título correspondiente. En cada encabezado de columna aparecerá un nombre o frase que describa su contenido (y no el nombre del campo).
- 2) Obtener una lista de los campos que son clave primaria en la base de datos pubs.
- 3) Obtener una lista que contenga tanto la descripción de todos los posibles tipos de empleos, como sus niveles máximos y mínimos. Se presentará ordenada descendientemente según el nivel máximo y ascendientemente según la descripción.
- 4) Obtener una lista que contenga los tipos de empleos, con sus niveles máximo y mínimo, y cuyo nivel máximo no sea mayor a 200, ordenados de forma descendente según su nivel máximo.
- 5) Obtener una lista de todos los pedidos realizados el 14 de Septiembre de 1994 de manera que aparezca ordenada por el identificador de título y se muestre la misma información que en el ejercicio (1).
- 6) Obtener una lista que contenga todos los tipos de empleos, con todos sus datos asociados, excepto los que tienen de identificador el 1, 3, 4, 7, 8 y 13.
- 7) Obtener una lista de todos los pedidos excepto los realizados en Septiembre de 1994.
- 8) Obtener una lista de los libros cuyo tema trate sobre bases de datos o sobre ordenadores y que estén clasificados en la categoría de “negocios”.
- 9) Obtener la misma lista que antes, limitando el título a 20 caracteres y el texto de las notas a 45 caracteres. En los encabezados de cada columna aparecerá una palabra que describa su contenido.
- 10) Obtener un listado con todos los datos contenidos en las columnas identificador del título, tipo, precio, identificador del editor (pub\_id), royalties y ventas anuales.
- 11) Obtener el precio máximo de todos los libros de cocina (tanto la moderna como la tradicional).
- 12) Listado de los libros (su identificador), tipos y royalties, mostrando en una nueva columna un valor propuesto de 15 dólares para aquellos cuyo royalty sea nulo. Poner en cada columna un encabezado apropiado y se ordenará por la columna con royalties propuestos.
- 13) Obtener un listado de todas las categorías de libros, con su precio medio por categoría, indicando además cuál es el precio medio de los precios medios.



- 14) Obtener un listado de los títulos de los libros, su categorías y los adelantos percibidos por libro, indicando además el valor medio de los adelantos para cada categoría. Recortar el título para que sólo ocupe doce caracteres.
- 15) Obtener un listado de las categorías de los libros con el precio máximo y el mínimo por categoría mostrando solo aquellas cuyo precio máximo sea menor de \$25.
- 16) Obtener un listado de los nombres de las tiendas en el que aparezca los números de pedido de las ventas, la cantidad y el descuento que aplica esa tienda.
- 17) Obtener un listado como el anterior pero que además aparezcan los títulos de los libros asociados con cada pedido.
- 18) Obtener un listado como el anterior en el que además aparezca el precio unitario de cada libro y el importe total por pedido, antes de aplicarle el descuento.
- 19) Obtener un listado del nombre y apellido de los empleados cuyo empleo tenga un nivel máximo superior a 200. El nombre y el apellido van en la misma columna con un encabezado descriptivo.
- 20) Crear la tabla EMPLEADOS, que tiene los campos: DNI, Nombre, Apellidos, Dirección y Jefe. Es imprescindible introducir el Nombre y los Apellidos. La clave primaria es DNI y en Jefe se pone el DNI del empleado que es su jefe.
- 21) a) Insertar en la tabla títulos un nuevo libro titulado “Placas base”, publicado el seis de Octubre de 2000, perteneciente a la categoría hardware y con clave HA0001.  
b) Insertar en la misma tabla otro libro con idénticas características al anterior, excepto que se titula “Memorias caché”. Indicar qué mensaje da el servidor.  
c) Cambiar la clave del registro anterior y volver a insertarlo.
- 22) a) Borrar los dos registros anteriores con una sola instrucción.  
b) Insertarlos nuevamente poniendo como fecha de publicación el 10 de Junio de 1991.  
c) Obtener un listado de los títulos publicados en Junio de 1991.
- 23) Crear un procedimiento que devuelva el precio de un libro dado, tras pasarle como parámetro su identificador de precio. Si no se le pasa ningún parámetro devolverá una lista con todos los identificadores con sus precios, ordenados descendientemente por precio
- 24) Crear un procedimiento que devuelva el precio de un libro dado, tras pasarle como parámetro su identificador de precio. Si no se le pasa ningún parámetro devolverá el precio medio de todos los libros.
- 25) Crear un procedimiento que admita dos parámetros, un identificador de título y un precio. Si no se le pasa el identificador de título muestra un mensaje indicando este error y termina la ejecución. Cuando se le pasan un identificador

de título y un precio cambia el precio al libro poniendo el parámetro como nuevo valor. Posteriormente muestra el valor del precio y el identificador.

- 26) Suponiendo que en la tabla TITLES existiera un campo llamado “venacu” (ventas acumuladas), crear un disparador para la tabla SALES, de manera que cada vez que se vende un libro incremente el valor de dicho campo en la cantidad correspondiente.

## 12. SOLUCIONES A LOS EJERCICIOS PROPUESTOS.

1)

```
SELECT ord_num "Número de pedido", ord_date "Fecha de
        pedido", title_id "Id de título"
FROM sales
```

| Número de pedido | Fecha de pedido         | Id de título |
|------------------|-------------------------|--------------|
| 6871             | 1994-09-14 00:00:00.000 | BU1032       |
| 722a             | 1994-09-13 00:00:00.000 | PS2091       |
| A2976            | 1993-05-24 00:00:00.000 | PC8888       |
| .....            | .....                   | .....        |
| 423LL922         | 1994-09-14 00:00:00.000 | MC3021       |
| 423LL930         | 1994-09-14 00:00:00.000 | BU1032       |
| P723             | 1993-03-11 00:00:00.000 | BU1111       |
| QA879.1          | 1993-05-22 00:00:00.000 | PC1035       |

(21 filas afectadas)

2)

```
SELECT name
FROM sysobjects
WHERE type="k"
```

```
name
-----
UPKCL_auidind
UPKCL_pubind
UPKCL_titleidind
UPKCL_taind
UPK_storeid
UPKCL_sales
PK__jobs__22AA2996
UPKCL_pubinfo
PK_emp_id
pk_dtproperties
(10 filas afectadas)
```

3)

```
SELECT job_desc, min_lvl, max_lvl
FROM jobs
ORDER BY max_lvl desc, job_desc
```

| job_desc                     | min_lvl | max_lvl |
|------------------------------|---------|---------|
| Chief Executive Officer      | 200     | 250     |
| Chief Financial Officer      | 175     | 250     |
| Publisher                    | 150     | 250     |
| Business Operations Manager  | 175     | 225     |
| Managing Editor              | 140     | 225     |
| .....                        |         |         |
| Designer                     | 25      | 100     |
| Editor                       | 25      | 100     |
| Sales Representative         | 25      | 100     |
| New Hire - Job not specified | 10      | 10      |

(14 filas afectadas)

4)

```
SELECT job_desc, min_lvl, max_lvl
FROM jobs
WHERE max_lvl != 200
ORDER BY max_lvl desc
```

| job_desc                     | min_lvl | max_lvl |
|------------------------------|---------|---------|
| Marketing Manager            | 120     | 200     |
| Public Relations Manager     | 100     | 175     |
| Acquisitions Manager         | 75      | 175     |
| Productions Manager          | 75      | 165     |
| Operations Manager           | 75      | 150     |
| Editor                       | 25      | 100     |
| Sales Representative         | 25      | 100     |
| Designer                     | 25      | 100     |
| New Hire - Job not specified | 10      | 10      |

(9 filas afectadas)

5)

```
SELECT ord_num "Número de pedido", ord_date "Fecha de
           pedido", title_id "Id de título"
FROM sales
WHERE ord_date >= '14/09/1994' and ord_date < '15/9/1994'
```

| Número de pedido | Fecha de pedido         | Id de título |
|------------------|-------------------------|--------------|
| -----            | -----                   | -----        |
| 6871             | 1994-09-14 00:00:00.000 | BU1032       |
| D4482            | 1994-09-14 00:00:00.000 | PS2091       |
| N914008          | 1994-09-14 00:00:00.000 | PS2091       |
| N914014          | 1994-09-14 00:00:00.000 | MC3021       |
| 423LL922         | 1994-09-14 00:00:00.000 | MC3021       |
| 423LL930         | 1994-09-14 00:00:00.000 | BU1032       |

(6 filas afectadas)

6)

```
SELECT job_id, job_desc, min_lvl, max_lvl
FROM jobs
WHERE job_id not in ("1", "3", "4", "7", "8", "13")
```

| job_id | job_desc                | min_lvl | max_lvl |
|--------|-------------------------|---------|---------|
| -----  | -----                   | -----   | -----   |
| 2      | Chief Executive Officer | 200     | 250     |
| 5      | Publisher               | 150     | 250     |
| 6      | Managing Editor         | 140     | 225     |
| 9      | Acquisitions Manager    | 75      | 175     |
| 10     | Productions Manager     | 75      | 165     |
| 11     | Operations Manager      | 75      | 150     |
| 12     | Editor                  | 25      | 100     |
| 14     | Designer                | 25      | 100     |

(8 filas afectadas)

7)

```
SELECT ord_num "Número de pedido", ord_date "Fecha de
           pedido", title_id "Id de título"
FROM sales
WHERE ord_date >= '1/10/1994' or ord_date < '1/9/1994'
```

| Número de pedido | Fecha de pedido         | Id de título |
|------------------|-------------------------|--------------|
| -----            | -----                   | -----        |
| A2976            | 1993-05-24 00:00:00.000 | PC8888       |
| P2121            | 1992-06-15 00:00:00.000 | TC3218       |
| P2121            | 1992-06-15 00:00:00.000 | TC4203       |
| P2121            | 1992-06-15 00:00:00.000 | TC7777       |
| P3087a           | 1993-05-29 00:00:00.000 | PS1372       |
| P3087a           | 1993-05-29 00:00:00.000 | PS2106       |
| P3087a           | 1993-05-29 00:00:00.000 | PS3333       |
| P3087a           | 1993-05-29 00:00:00.000 | PS7777       |
| QQ2299           | 1993-10-28 00:00:00.000 | BU7832       |
| TQ456            | 1993-12-12 00:00:00.000 | MC2222       |

```

X999          1993-02-21 00:00:00.000      BU2075
P723          1993-03-11 00:00:00.000      BU1111
QA879.1       1993-05-22 00:00:00.000      PC1035
(13 filas afectadas)

```

**8)**

```

SELECT title, type ,notes
FROM TITLES
WHERE (notes like "%database%" or notes like "%computer%")
      and type="business"

```

| Title                               | type     | notes   |
|-------------------------------------|----------|---|
| The Busy Executive's Database Guide | business | An overview of available database systems with emphasis on common business applications. Illustrated. |
| Straight Talk About Computers       | business | Annotated analysis of what computers can do for you: a no-hype guide for the critical user.           |

(2 filas afectadas)

**9)**

```

SELECT  convert(varchar(20),title) Título, type
        Tipo,convert(varchar(45),notes) Notas
FROM TITLES
WHERE (notes like "%database%" or notes like "%computer%")
      and type="business"

```

| Título               | Tipo     | Notas   |
|----------------------|----------|---|
| The Busy Executive's | business | An overview of available database systems wit |
| Straight Talk About  | business | Annotated analysis of what computers can do f |

(2 filas afectadas)

**10)**

```

SELECT title_id, type, price, pub_id, royalty, ytd_sales
FROM titles

```

| title_id | type     | price   | pub_id | royalty | ytd_sales |
|----------|----------|---------|--------|---------|-----------|
| BU1032   | business | 19.9900 | 1389   | 10      | 4095      |

|        |              |         |      |      |       |
|--------|--------------|---------|------|------|-------|
| BU1111 | business     | 11.9500 | 1389 | 10   | 3876  |
| BU2075 | business     | 2.9900  | 0736 | 24   | 18722 |
| BU7832 | business     | 19.9900 | 1389 | 10   | 4095  |
| MC2222 | mod_cook     | 19.9900 | 0877 | 12   | 2032  |
| MC3021 | mod_cook     | 2.9900  | 0877 | 24   | 22246 |
| MC3026 | UNDECIDED    | NULL    | 0877 | NULL | NULL  |
| PC1035 | popular_comp | 22.9500 | 1389 | 16   | 8780  |
| PC8888 | popular_comp | 20.0000 | 1389 | 10   | 4095  |
| PC9999 | popular_comp | NULL    | 1389 | NULL | NULL  |
| PS1372 | psychology   | 21.5900 | 0877 | 10   | 375   |
| PS2091 | psychology   | 10.9500 | 0736 | 12   | 2045  |
| PS2106 | psychology   | 7.0000  | 0736 | 10   | 111   |
| PS3333 | psychology   | 19.9900 | 0736 | 10   | 4072  |
| PS7777 | psychology   | 7.9900  | 0736 | 10   | 3336  |
| TC3218 | trad_cook    | 20.9500 | 0877 | 10   | 375   |
| TC4203 | trad_cook    | 11.9500 | 0877 | 14   | 15096 |
| TC7777 | trad_cook    | 14.9900 | 0877 | 10   | 4095  |

**11)**

```
SELECT max(price)
FROM titles
WHERE type='mod_cook' or type='trad_cook'
```

```
-----
20.9500
(1 filas afectadas)
```

**12)**

```
SELECT title_id Identificador, type Tipo, royalty Royalties,
       isnull(royalty, $15) "Royalties propuestos"
FROM titles
ORDER BY 4
```

| Identificador | Tipo         | Royalties | Royalties propuestos |
|---------------|--------------|-----------|----------------------|
| -----         |              |           |                      |
| BU1032        | business     | 10        | 10                   |
| PC8888        | popular_comp | 10        | 10                   |
| .....         |              |           |                      |
| PS1372        | psychology   | 10        | 10                   |
| MC2222        | mod_cook     | 12        | 12                   |
| .....         |              |           |                      |
| PS2091        | psychology   | 12        | 12                   |
| TC4203        | trad_cook    | 14        | 14                   |
| MC3026        | UNDECIDED    | NULL      | 15                   |
| PC9999        | popular_comp | NULL      | 15                   |

```

PC1035      popular_comp 16      16
BU2075      business    24      24
MC3021      mod_cook    24      24
(18 filas afectadas)

```

**13)**

```

SELECT type, avg(price)
FROM titles
GROUP BY type
COMPUTE avg(avg(price))

```

```

type
-----
business      13.7300
mod_cook      11.4900
popular_comp  20.9833
psychology    13.5040
trad_cook     15.9633
UNDECIDED     NULL

          avg
          =====
          15.1341
(7 filas afectadas)

```

Advertencia. Valor nulo eliminado del agregado.

**14)**

```

SELECT convert(char(12),title), type, advance
FROM titles
ORDER BY type
COMPUTE avg(advance) BY type

```

```

          type      advance
-----
The Busy Exe business    5000.0000
Cooking with business    5000.0000
You Can Comb business    10125.0000
Straight Tal business    5000.0000
          avg
          =====
          6281.2500

Silicon Vall mod_cook     .0000
The Gourmet mod_cook     15000.0000
          avg
          =====
          7500.0000

But Is It Us popular_comp 7000.0000
Secrets of S popular_comp 8000.0000
Net Etiquett popular_comp NULL

```

```

                                avg
                                =====
                                7500.0000

Computer Pho psychology      7000.0000
.....
Emotional Se psychology     4000.0000
                                avg
                                =====
                                4255.0000

Onions, Leek trad_cook      7000.0000
Fifty Years trad_cook      4000.0000
Sushi, Anyon trad_cook     8000.0000
                                avg
                                =====
                                6333.3333

The Psycholo UNDECIDED     NULL
                                avg
                                =====

(24 filas afectadas)

```

Advertencia. Valor nulo eliminado del agregado.

### 15)

```

SELECT type, max(price),min(price)
FROM titles
GROUP BY type
HAVING max(price)<$25

```

```

type
-----
business      19.9900          2.9900
mod_cook      19.9900          2.9900
popular_comp  22.9500          20.0000
psychology    21.5900           7.0000
trad_cook     20.9500          11.9500
(5 filas afectadas)

```

Advertencia. Valor nulo eliminado del agregado.

### 16)

```

SELECT st.stor_name,s.ord_num, s.qty, d.discount
FROM stores st JOIN sales s on st.stor_id=s.stor_id
JOIN discounts d on d.stor_id=st.stor_id

```

```

stor_name  ord_num      qty    discount
-----
Bookbeat   423LL922      15      5.00
Bookbeat   423LL930      10      5.00

```



|          |         |    |      |
|----------|---------|----|------|
| Bookbeat | P723    | 25 | 5.00 |
| Bookbeat | QA879.1 | 30 | 5.00 |

17)

```
SELECT st.stor_name,s.ord_num,convert(char(20),t.title)
       Título, s.qty, d.discount
FROM stores st JOIN sales s on st.stor_id=s.stor_id
JOIN discounts d on d.stor_id=st.stor_id
JOIN titles t on t.title_id=s.title_id
```

| stor_name | ord_num  | Título               | qty | discount |
|-----------|----------|----------------------|-----|----------|
| Bookbeat  | 423LL922 | The Gourmet Microwav | 15  | 5.00     |
| Bookbeat  | 423LL930 | The Busy Executive's | 10  | 5.00     |
| Bookbeat  | P723     | Cooking with Compute | 25  | 5.00     |
| Bookbeat  | QA879.1  | But Is It User Frien | 30  | 5.00     |

18)

```
SELECT st.stor_name,s.ord_num,convert(char(20),t.title)
       Título, t.price Precio, s.qty, d.discount,
       (price*qty) "Importe total"
FROM stores st JOIN sales s on st.stor_id=s.stor_id
JOIN discounts d on d.stor_id=st.stor_id
JOIN titles t on t.title_id=s.title_id
```

| stor_name | ord_num  | Título               | Precio  | qty | discount | Importe total |
|-----------|----------|----------------------|---------|-----|----------|---------------|
| Bookbeat  | 423LL922 | The Gourmet Microwav | 2.9900  | 15  | 5.00     | 44.8500       |
| Bookbeat  | 423LL930 | The Busy Executive's | 19.9900 | 10  | 5.00     | 199.9000      |
| Bookbeat  | P723     | Cooking with Compute | 11.950  | 25  | 5.00     | 298.7500      |
| Bookbeat  | QA879.1  | But Is It User Frien | 22.9500 | 30  | 5.00     | 688.5000      |

19)

```
SELECT fname + " " + lname
FROM employee
WHERE job_id IN
      (SELECT job_id
       FROM jobs
       WHERE max_lvl > $200)
```

```
-----  
Victoria Ashworth  
Francisco Chang  
Philip Cramer  
Ann Devon  
Paul Henriot  
Carlos Hernadez  
Matti Karttunen  
Janine Labrune  
Laurence Lebihan  
Rita Muller  
Sven Ottlieb  
Maria Pontes  
Diego Roel  
Annette Roulet  
(14 filas afectadas)
```

**20)**

```
CREATE TABLE EMPLEADO(  
EmpDNI char(10) not null constraint PK_C_DNIEmp primary key,  
EmpNombre varchar(15) not null,  
EmpApellidos varchar (30) not null,  
EmpDirecc varchar (50),  
EmpJefe cah(10) constraint FK_NC_Empleado_Jefe referentes  
EMPLEADO)
```

**21)**

```
a)  
INSERT TITLES (title_id, title, type, pubdate)  
VALUES ('HA0001', 'Placas base', 'hardware','6/10/2000')
```

```
b)  
INSERT TITLES (title_id, title, type, pubdate)  
VALUES ('HA0001', 'Memorias caché', 'hardware','6/10/2000')
```

```
Servidor: mensaje 2627, nivel 14, estado 1, línea 1  
Infracción de la restricción PRIMARY KEY 'UPKCL_titleidind'.  
No se puede insertar una clave duplicada en el objeto  
'titles'.  
Se terminó la instrucción.
```

```
c)  
INSERT TITLES (title_id, title, type, pubdate)  
VALUES ('HA0002', 'Memorias caché', 'hardware','6/10/2000')
```

**22)**

a)

```
DELETE TITLES WHERE title_id='HA0001' OR title_id='HA0002'
```

b)

```
INSERT TITLES (title_id, title, type, pubdate)
```

```
VALUES ('HA0001', 'Placas base', 'hardware', '10/6/1991')
```

```
INSERT TITLES (title_id, title, type, pubdate)
```

```
VALUES ('HA0002', 'Memorias caché', 'hardware', '10/6/1991')
```

c)

```
SELECT title_id, title, type, pubdate
```

```
FROM titles
```

```
WHERE pubdate >= '1/6/1991' AND pubdate < '1/7/1991'
```

| title_id |              | type         | pubdate                 |
|----------|--------------|--------------|-------------------------|
| BU1032   | The Busy Exe | business     | 1991-06-12 00:00:00.000 |
| BU1111   | Cooking with | business     | 1991-06-09 00:00:00.000 |
| BU2075   | You Can Comb | business     | 1991-06-30 00:00:00.000 |
| BU7832   | Straight Tal | business     | 1991-06-22 00:00:00.000 |
| HA0001   | Placas base  | hardware     | 1991-06-10 00:00:00.000 |
| HA0002   | Memorias cac | hardware     | 1991-06-10 00:00:00.000 |
| MC2222   | Silicon Vall | mod_cook     | 1991-06-09 00:00:00.000 |
| MC3021   | The Gourmet  | mod_cook     | 1991-06-18 00:00:00.000 |
| PC1035   | But Is It Us | popular_comp | 1991-06-30 00:00:00.000 |
| PS2091   | Is Anger the | psychology   | 1991-06-15 00:00:00.000 |
| PS3333   | Prolonged Da | psychology   | 1991-06-12 00:00:00.000 |
| PS7777   | Emotional Se | psychology   | 1991-06-12 00:00:00.000 |
| TC4203   | Fifty Years  | trad_cook    | 1991-06-12 00:00:00.000 |
| TC7777   | Sushi, Anyon | trad_cook    | 1991-06-12 00:00:00.000 |

(14 filas afectadas)

**23)**

```
create proc prPrecioID10
```

```
(@title_id char(6)=null)
```

```
as
```

```
if @title_id is null
```

```
begin
```

```
    SELECT title_id Identificador, price Precio
```

```
    FROM titles
```

```
    ORDER BY price desc
```

```
    return
```

```
end
```

```
SELECT title_id, price
```

```
FROM titles
WHERE title_id=@title_id
return
```

```
exec prPrecioID10
```

```
Identificador Precio
-----
```

|        |         |
|--------|---------|
| PC1035 | 22.9500 |
| PS1372 | 21.5900 |
| TC3218 | 20.9500 |
| PC8888 | 20.0000 |
| BU1032 | 19.9900 |
| BU7832 | 19.9900 |
| MC2222 | 19.9900 |
| PS3333 | 19.9900 |
| TC7777 | 14.9900 |
| BU1111 | 11.9500 |
| TC4203 | 11.9500 |
| PS2091 | 10.9500 |
| PS7777 | 7.9900  |
| PS2106 | 7.0000  |
| BU2075 | 2.9900  |
| MC3021 | 2.9900  |
| MC3026 | NULL    |
| PC9999 | NULL    |

(18 filas afectadas)

## 24)

```
create proc prPrecioID11
(@title_id char(6)=null)
as
if @title_id is null
begin
    SELECT avg(price) "Precio Medio"
    FROM titles
    return
end
SELECT title_id, price
FROM titles
WHERE title_id=@title_id
return
```

```
exec prPrecioID11
```

```
Precio Medio
-----
14.7662
(1 filas afectadas)
```

Advertencia. Valor nulo eliminado del agregado.

25)

```

create proc prcampre
(@title_id char(6)=null,
@price money = null)
as
if @title_id is null
begin
    print 'El procedimiento prcampre requiere el parámetro
    title_id'
    return
end
if @price is not null
begin
    update titles
    set price=@price
    where title_id=@title_id
    if @@rowcount = 1
        print 'precio cambiado'
end
SELECT title_id, price
FROM titles
WHERE title_id=@title_id
return

```

**Ejecución:**

```
prcampre PC9999, 20
```

**Resultado:**

```

precio cambiado
title_id price
-----
PC9999    20.0000
(1 filas afectadas)

```

26)

```

create trigger trVentasIncre
on VENTAS for insert
as
UPDATE TITULOS
SET TITULOS.venacu= TITULOS.venacu + (inserted.qty*
                                     TITULOS.price)
FROM TITULOS, inserted
WHERE TITULOS.title_id=inserted.title_id
print "Tabla titulos actualizada por el disparador"
return

```

**Para probarlo:**

```

INSERT VENTAS stor_id,ord_num,ord_date,qty,payterms,title_id
VALUES ('9002','K111','1/1/2001',4,'Visa','NE1113')

```



# TEMA 6. ÍNDICES Y FUNCIONES DE ASOCIACIÓN

## 1. INTRODUCCIÓN.

### 1.1. SITUACIÓN.

La estructura física que almacena los datos de una base de datos es una fila compuesta por las diferentes tablas; éstas, a su vez, se dividen en archivos, y éstos en registros. De tal forma que para acceder a un registro hay que buscarlo dentro del archivo correspondiente.

La búsqueda de un registro que contenga un campo que satisfaga una condición se puede realizar de dos formas:

- **Secuencial:** es más sencilla ya que no necesita utilizar estructuras adicionales de datos. Sin embargo es un método lento, y por tanto poco eficiente. Su uso es adecuado en la búsqueda entre un pequeño número de objetos. Se pierde mucho tiempo en tablas grandes, ya que se recorre la tabla hasta encontrarlo.
- **Directa:** lo más probable es que una búsqueda no pueda ser totalmente directa. Consiste en tener un puntero que, dada una clave de búsqueda, indique el registro deseado. Su principal ventaja es la rapidez y la eficacia. El inconveniente es que su uso es un método un poco artificioso, es necesario utilizar una estructura de datos adicional y, como consecuencia de esto, la necesidad de más espacio. Su uso es conveniente cuando el número de objetos supere el centenar.

La **clave de búsqueda** es un atributo o conjunto de atributos utilizados para buscar un registro en un conjunto de ellos. No tiene por qué coincidir con la clave primaria.

❖ Ejemplo: Libros de la categoría “novela histórica”.

El catálogo de la biblioteca tiene las fichas ordenadas alfabéticamente (títulos, autores, materias). Ejemplos de búsquedas:

- Buscar un libro por su autor:  
Si ese autor tiene un único libro → directa.  
Si hay varios libros (de ese autor o de otros autores con el mismo apellido) → directa hasta autor, luego secuencial entre los libros del mismo autor.
- Materias: directa hasta materia, luego secuencial.
- Títulos: directa hasta la primera letra del título, luego secuencial.

Hay que tener cuidado en definir correctamente dónde se deja la búsqueda directa y se toma la secuencial.

## 1.2. MÉTODOS DE ACCESO BÁSICOS.

- Métodos;
  - a) Índices ordenados: son índices basados en una disposición ordenada de los valores.
  - b) Índices asociativos: son índices basados en una distribución uniforme de los valores a lo largo de una serie de cajones. A cada cajón se le asigna un valor determinado por una función (función de asociación = *hash function*).
- Criterios de valoración;
  - a) Tipos de acceso soportados: hay índices que soportan bien las búsquedas en las que se quiere que un registro cumpla una determinada condición (búsqueda por valor). Otros índices soportan mejor un conjunto de valores (búsqueda por rango). Según el tipo de acceso se utilizará un índice u otro.
  - b) Tiempo de acceso a los datos.
  - c) Tiempo de inserción: al insertar un registro en la tabla en ocasiones hay que insertar una clave de búsqueda o índice en la estructura de índices.
  - d) Tiempo de borrado: al borrar un registro, habrá que borrar el índice asociado a él, dependiendo de si ese índice lo utiliza otro registro o no.
  - e) Espacio adicional requerido: el espacio no suele ser un factor determinante, pero puede que interese que ocupe poco en memoria principal, que quepa en ella, que haya pocos accesos, etc.

## 2. ÍNDICES ORDENADOS.

Almacenan de manera ordenada los valores de las claves de búsqueda y asocian a cada clave los registros que contienen esa clave de búsqueda. (Por ejemplo, en una biblioteca se tendría el orden por títulos, por autores, por materias, etc.).



Una tabla (conjunto de registros que se almacenan en los archivos) puede tener varios índices (esto suele ser muy común); cada uno de ellos es para una clave de búsqueda distinta.

Consisten en una estructura de datos ordenados secuencialmente que almacena las claves de búsqueda de la estructura de índices. La entrada de índice, es decir, la fila de la estructura de índices, está compuesta por dos campos:

- la clave de búsqueda;
- un puntero a los datos.

## 2.1. ÍNDICE PRIMARIO.

Cuando los registros se encuentran ordenados dentro de un archivo según una clave de búsqueda determinada, el índice asociado a esa clave de búsqueda recibe el nombre de índice primario. El índice primario es un índice agrupado (*clustering index*) y **no** tiene que corresponder necesariamente con la clave primaria de una tabla, aunque es muy habitual que esto ocurra.

La clave primaria es el índice de la tabla por defecto. Si está agrupado (*clustered*), los valores de la tabla se ordenan físicamente por ese valor de clave primaria. Si el índice coincide con la clave primaria es el índice primario, que está ordenado y agrupado.

Un **índice** es la estructura de datos que permite acceder a los archivos o datos almacenados.

A los archivos ordenados secuencialmente según una clave de búsqueda se les llama **archivos secuenciales indexados**.

|         |       |            |           |
|---------|-------|------------|-----------|
| Blanes  | Ana   | Body Pump  | 10-1-1999 |
| Caño    | Juan  | Step       | 13-3-2001 |
| Caño    | Luis  | Spinning   | 20-6-2001 |
| Fuertes | José  | Step       | 10-1-2002 |
| López   | Ivan  | Step       | 12-6-2000 |
| López   | Inés  | Body Pump  | 24-4-2000 |
| López   | Eva   | Cardio box | 13-3-2001 |
| Montes  | Diana | Spinning   | 13-3-1999 |
| Mures   | Rosa  | Step       | 14-6-2000 |

El índice primario es la primera columna.

### 2.1.1. Índices densos y dispersos.

Hay dos tipos de índices ordenados:

- Índice denso: contiene un registro índice para cada valor diferente de la clave de búsqueda de la tabla.
- Índice disperso: contiene un registro índice sólo para algunos valores de la clave de búsqueda.

A lo máximo, en el índice habrá tantos valores distintos como haya en la tabla.

#### Densidad de un índice:

$$d = \frac{\text{número de valores de la clave en el índice}}{\text{número de valores de clave diferentes}}; \quad d \in [0,1]$$

Si  $d = 1$ : es un índice denso y contiene todas las claves.

Si  $d < 1$ : es un índice disperso y no contiene todas las claves.

#### Denso:

|         |   |   |         |       |           |           |
|---------|---|---|---------|-------|-----------|-----------|
| Blanes  | • | → | Blanes  | Ana   | Body Pump | 20-1-1999 |
| Caño    | • | → | Caño    | Juan  | Step      | 13-3-2001 |
| Fuertes | • | → | Caño    | Luis  | Spinning  | 20-6-2001 |
| López   | • | → | Fuertes | José  | Step      | 10-1-2002 |
| Montes  | • | → | López   | Iván  | Step      | 12-6-2000 |
| Mures   | • | → | López   | Inés  | Body Pump | 24-4-2000 |
|         |   | → | López   | Eva   | Cardiobox | 13-3-2001 |
|         |   | → | Montes  | Diana | Spinning  | 13-3-1999 |
|         |   | → | Mures   | Rosa  | Step      | 14-6-2000 |

#### Disperso:

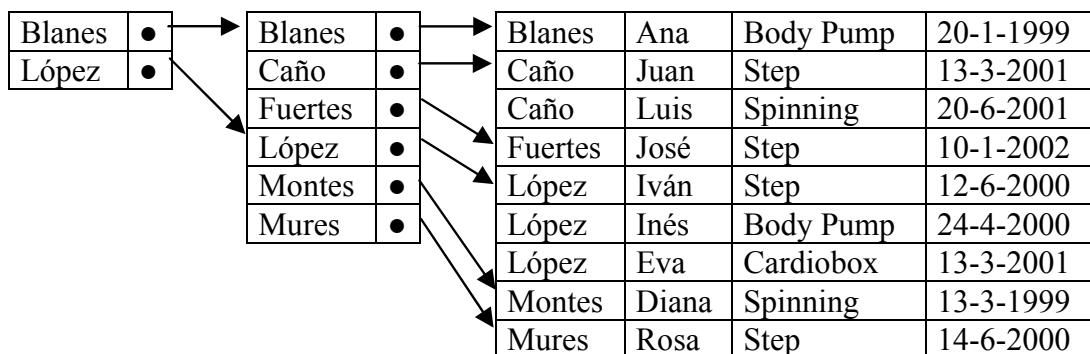
|         |   |   |         |       |           |           |
|---------|---|---|---------|-------|-----------|-----------|
| Blanes  | • | → | Blanes  | Ana   | Body Pump | 20-1-1999 |
| Fuertes | • | → | Caño    | Juan  | Step      | 13-3-2001 |
| Montes  | • | → | Caño    | Luis  | Spinning  | 20-6-2001 |
|         |   | → | Fuertes | José  | Step      | 10-1-2002 |
|         |   | → | López   | Iván  | Step      | 12-6-2000 |
|         |   | → | López   | Inés  | Body Pump | 24-4-2000 |
|         |   | → | López   | Eva   | Cardiobox | 13-3-2001 |
|         |   | → | Montes  | Diana | Spinning  | 13-3-1999 |
|         |   | → | Mures   | Rosa  | Step      | 14-6-2000 |

- Búsqueda de valores mediante el uso de índices dispersos:  
Se busca la entrada del índice con el mayor valor que sea igual o menor que el valor buscado. Se lee el valor de la clave al que apunta esa entrada del índice y se sigue buscando secuencialmente en el archivo hasta encontrar el registro deseado.

- Ventajas e inconvenientes:
  - Los índices densos son más rápidos porque no hay que hacer esa búsqueda secuencial en la estructura del índice, aunque los dispersos necesitan menos memoria y menos tiempo en inserciones y borrados (tiempo de actualización).
  - Es necesario, por tanto, buscar un compromiso entre el tiempo de acceso y el espacio adicional requerido.
  - Un índice disperso con una entrada por cada página en disco duro ofrece unos buenos resultados.

### 2.1.2. Índices multinivel o jerarquizados.

- Índice y memoria principal.  
Idealmente un índice será una estructura de datos suficientemente pequeña como para alojarse completamente en memoria principal. Si es demasiado grande será necesario acceder a disco, lo cual es más costoso.  
En una página, con el tamaño actual, caben solo tres registros de índice.  
Para evitar excesivos accesos a disco se utilizan índices multinivel.
- Índices multinivel.  
Son estructuras de datos con índices para acceder a los elementos del índice del nivel inmediatamente inferior. Hay varios niveles de índices. El de nivel 0 es el más cercano a los datos. El de nivel 1 es más pequeño que el de nivel 0 y así sucesivamente. En general, se utilizan índices de un cierto tamaño, de forma que entre completamente en una página de la memoria principal, para acceder así a bloques de tamaño una página de memoria en el nivel siguiente.  
Lógicamente, los índices de niveles superiores siempre serán índices dispersos para así tener menos entradas y acelerar el acceso a índices de menor nivel.  
Si el índice superior se hace demasiado grande se añade un nuevo nivel de indexación.



En el ejemplo, el primer índice es disperso y el segundo, denso.

### 2.1.3. Actualización de índices primarios.

Siempre que se inserta o elimina un registro de un archivo, hay que actualizar el índice. Los índices primarios pueden ser de dos tipos:

1) Índices de un nivel.

a) Algoritmo de borrado.

1.- Buscar el registro que se desea borrar y el índice asociado.

- Si no hay más registros con la misma clave de búsqueda, borrar el registro y eliminar la entrada del índice.
- En caso contrario, borrar el registro solamente.

2.- Borrado de la entrada del índice:

- Índice denso: Igual que borrar un registro de un archivo.
- Índice disperso: Se borra un valor clave sustituyendo su entrada (si existe) en el índice por el siguiente valor de la clave (según el orden de la clave de búsqueda).
- Si el siguiente valor de la clave tenía ya una entrada, eliminar la entrada del índice.

b) Algoritmo de inserción.

1.- Buscar la clave de búsqueda del nuevo registro en el índice.

- Si el valor de la clave no está en el índice:
  - » Índice denso: Insertar en el lugar adecuado la clave de búsqueda.
  - » Índice disperso: Si se almacena una entrada por página, no se modifica el índice. Si es necesario crear una nueva página, insertar en el índice el primer valor de la clave que aparezca en la nueva página.

2.- A continuación se inserta el registro en el archivo.

2) Índices jerarquizados.

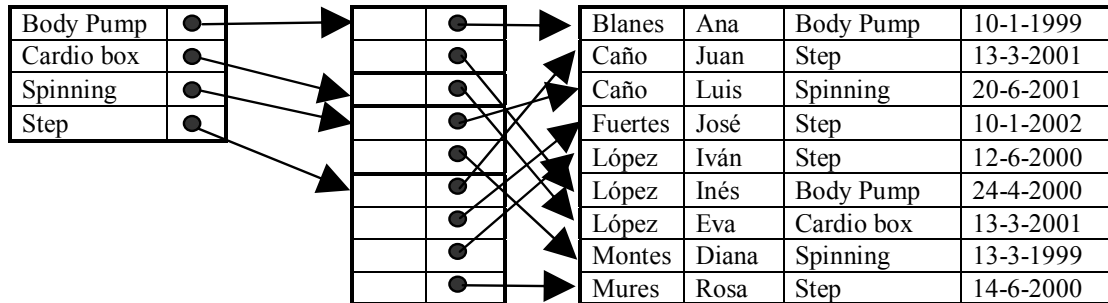
Siempre se actualiza en primer lugar el de nivel más bajo. Se trata como un archivo de registros y, en el supuesto de que se esté insertando un índice, será necesario actualizar de forma similar el índice superior (y así sucesivamente con todos los niveles).

### 2.2. ÍNDICES SECUNDARIOS (O NO AGRUPADOS).

Este tipo de índices se utiliza cuando se desean tener otros índices por tabla, diferentes del índice primario, ya que no sólo se quiere recuperar el valor de la clave primaria. Interesa crear un índice secundario para los campos que se consulten frecuentemente. Si se quiere puede estar formado por más de un campo. En esta ocasión, el archivo está ordenado físicamente según los valores del índice primario; por lo tanto:

- » El archivo no estará ordenado según las claves de búsqueda del índice secundario (es un índice no agrupado).
- » No se podrán utilizar índices dispersos.

- » El índice secundario utilizado será denso, y contendrá una entrada para cada valor de búsqueda, así como un puntero a cada registro del archivo. Se pueden utilizar estructuras intermedias.



### 2.3. ÁRBOLES EQUILIBRADOS.

El inconveniente que tiene el uso de los índices ordenados es la degradación del rendimiento en las búsquedas a medida que el tamaño de la base de datos aumenta. Esto conlleva una reorganización del archivo.

Los **árboles B+** son árboles balanceados que tienen en cada nodo del árbol más de dos punteros (tantos como quepan en una página de memoria) y que permiten que en la estructura del índice haya claves repetidas.

Los árboles B+ son más eficientes que los índices ordenados simples (son, por o general, más anchos y más bajos). Además, su eficiencia no se ve afectada por la inserción y borrado de datos. Sin embargo, requieren de un espacio adicional.

Este tipo de árboles está formado por una jerarquía de registros índices (nodos) junto con un archivo de registros de datos.

Cuando se trata de **árboles equilibrados** todas las hojas del árbol B+ están a la misma distancia del nodo raíz. Esto garantiza que el acceso a todos los registros (filas) de una tabla es igual de eficiente.

Los nodos intermedios apuntan a otros nodos, y las hojas, a los registros.

#### 2.3.1. Estructura de un nodo.

Un nodo se caracteriza por tener “n” punteros (cada uno denominado  $P_i$ ) y un máximo de “(n-1)” claves de búsqueda (denominadas  $K_i$ ).

Los punteros  $P_i$  apuntan a nodos de niveles inferiores.

Cuando el nodo es una hoja del árbol:

- »  $P_n$  apunta al siguiente nodo hoja (si existe); sino, no apunta a nada.
- » El resto de punteros  $P_i$  apuntan a un registro del archivo con clave de búsqueda  $K_i$  ( $i < n$ ), ó, si el archivo no está ordenado por clave y si la clave de búsqueda no es primaria, a un cajón de punteros apuntando cada uno a un registro con valor de la clave de búsqueda  $K_i$ .

|       |       |       |       |       |         |           |           |       |
|-------|-------|-------|-------|-------|---------|-----------|-----------|-------|
| $P_1$ | $K_1$ | $P_2$ | $K_2$ | $P_3$ | $\dots$ | $P_{n-1}$ | $K_{n-1}$ | $P_n$ |
|-------|-------|-------|-------|-------|---------|-----------|-----------|-------|

### 2.3.2. Propiedades de un árbol B+.

- I. El nodo raíz no tiene restricciones.
- II. Todos los caminos desde el nodo raíz hasta un nodo hoja tienen la misma longitud (árbol balanceado).
- III. Los nodos que no son raíz ni hoja contienen un número de punteros hacia nodos de niveles más bajos “p” tal que:

$$p \in \left[ \left\lceil \frac{n}{2} \right\rceil, n \right]$$

siendo n el número máximo de punteros, el cual se conoce cuando se crea el árbol.

- IV. Un nodo hoja tiene “h” punteros a registros del archivo de datos, de forma que:

$$h \in \left[ \left\lceil \frac{n-1}{2} \right\rceil, n-1 \right]$$

El puntero restante apunta al siguiente nodo, ya que todos están relacionados.

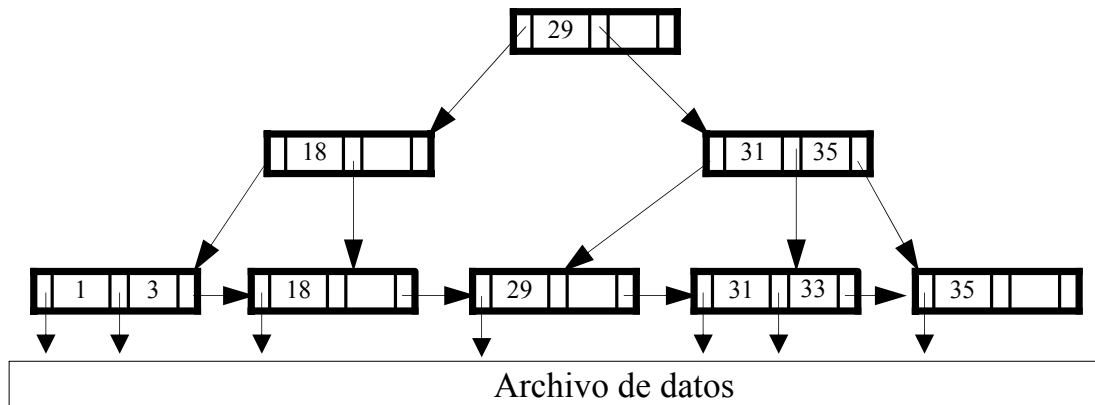
- V. En cada nodo, las claves siempre están ordenadas por valor ( $K_1 < K_2 \dots < K_{n-1}$ ).
- VI. Todas las claves del subárbol al que apunta  $P_1$  (el primer puntero de cada nodo) son estrictamente menores que  $K_1$ .
- VII. Todas las claves del subárbol al que apunte  $P_i$ ,  $2 \leq i \leq n-1$ , tienen valores dentro del rango  $[K_{i-1}, K_i]$ .
- VIII. Todas las claves del subárbol al que apunta  $P_n$  son mayores o iguales que  $K_{n-1}$ .

De VII y VIII se deduce que para  $i > 1$ ,  $K_{i-1}$  es siempre la clave más baja en el subárbol referido por  $P_i$ .

### 2.3.3. Consultas con árboles B+.

La búsqueda de los registros cuya clave de búsqueda sea  $K_x$  en árboles B+ se realiza mediante el siguiente algoritmo:

- 1) Buscar en el nodo raíz el menor valor,  $K_i$ , de la clave de búsqueda mayor que  $K_x$ .
  - Si este valor es  $K_i$ , seguir el puntero  $P_i$  al siguiente nivel del árbol.
  - En el caso de que haya “m-1” claves de búsqueda en el nodo ( $m \leq n$ ),
    - Si  $K_x \geq K_{m-1}$ , entonces seguir  $P_m$  al nodo que indique.
- 2) Repetir el paso 1 hasta llegar a un nodo hoja y a un puntero  $P_i$  que apunta al registro o cajón de punteros cuya clave de búsqueda es  $K_x$ .



Los punteros que apuntan a nodos, apuntan al nodo en general, no a una clave en concreto.

La eficiencia de los árboles B+ frente a los binarios se puede resumir con las siguientes ecuaciones, con “k” valores para la clave de búsqueda y siendo “l” la longitud del camino:

$$l_{B+} \leq \left\lceil \log_{\frac{n}{2}} K \right\rceil \quad \{\text{en árboles balanceados} +\}$$

$$l_B \leq \left\lceil \log_2 K \right\rceil \quad \{\text{en árboles binarios equilibrados}\}$$

❖ Ejemplo:

$K = 1.000.000$  y  $n = 100$

$l_{B+} = 4$

$l_B = 20$

NOTA: un árbol B+ se diferencia de otros árboles, como los binarios, en el uso de memoria y principalmente en el tamaño de un nodo y, por lo tanto, en la altura del árbol.

En un árbol binario, cada nodo es pequeño y tiene como máximo dos punteros. En un árbol B+ cada nodo es grande, normalmente del tamaño de una página de memoria de disco, y suele tener un gran número de punteros. Por ello son anchos y bajos en lugar de altos y estrechos como los binarios.

#### 2.3.4. Actualizaciones en árboles B+.

a) Inserción.

El algoritmo de inserción es el siguiente:

- Buscar el nodo hoja donde tendría que encontrarse el valor de la clave de búsqueda.
- Si la clave está ya en ese nodo, insertar el nuevo registro en el archivo y, si es necesario, un puntero en el cajón de punteros si el archivo no está ordenado por la clave primaria.
- Si no, insertar el valor en el nodo hoja de forma que las claves de búsqueda permanezcan ordenadas. Insertar ahora el

nuevo registro en el archivo y, si es necesario, se crea un cajón de punteros con un puntero apropiado. (Si al insertar hay que crear nuevos nodos en el árbol se crean cumpliendo las propiedades de los árboles balanceados).

Este algoritmo está pensado para mover el menor número de nodos posible; no importa el espacio ocupado.

b) Borrado.

- Buscar el registro que hay que borrar y eliminarlo del archivo.
- Si no hay un cajón asociado a la clave de búsqueda o si éste se vacía tras el borrado, hay que borrar el valor de la clave de búsqueda.

‡ Observaciones:

- Si al insertar es necesario dividir un nodo en dos, entonces:
  - Recolocar las claves para que  $\left\lceil \frac{n}{2} \right\rceil$  claves queden en el nodo izquierdo y el resto en el derecho.
  - Hay que actualizar las claves y los punteros en el nodo correspondiente del nivel superior. Puede ser necesario añadir un nuevo nodo.
- Si al borrar un nodo queda demasiado pequeño (menos de  $\left\lceil \frac{n}{2} \right\rceil$  punteros), será necesario combinar nodos para que ninguno tenga menos de  $\left\lceil \frac{n}{2} \right\rceil$  punteros.

### 3. ÍNDICES ASOCIATIVOS (HASHING).

Una de las desventajas de los índices ordenados es la necesidad que tienen de leer y acceder a estructuras de datos para localizar un registro. Por eso se utilizan las técnicas *hashing*, que eliminan la necesidad de esas estructuras de datos utilizando funciones de cálculo para localizar registros a partir de su clave de búsqueda. Son funciones que, dado el valor de la clave, devuelven directamente dónde está el registro.

#### 3.1. ASOCIACIÓN ESTÁTICA.

Una asociación estática es aquella que permanece invariable al modificarse el tamaño de la base de datos.

##### 3.1.1. Concepto de cajón.

Unidad de almacenamiento que puede alojar uno o más valores de la clave de búsqueda con sus respectivos punteros a registro.



Un cajón es una clave más un puntero o un conjunto de registros que contienen punteros a las direcciones de los registros del archivo de datos.

### 3.1.2. Función de asociación (*hash*).

La función de asociación se define de la siguiente forma:

Sea  $K$  el conjunto de todos los valores de la clave de búsqueda, sea  $B$  el conjunto de direcciones del cajón y sea la función de asociación  $h: K \rightarrow B$ ; entonces, el cajón en el que se almacena un registro de clave  $K_i$  viene dado por la función de asociación  $h(K_i)$ .

Una vez que se sabe el cajón que corresponde a una clave de búsqueda:

- a) Si hay que insertarla y hay sitio, se inserta allí.
- b) Si hay que buscarla, se busca dentro del cajón la clave de búsqueda deseada.
- c) Si hay que borrarla, se busca dentro del cajón esa clave de búsqueda y se elimina.

Las condiciones de diseño de una función de asignación son:

- a) Distribución uniforme: cada cajón ha de tener asociado el mismo número de valores de la clave de búsqueda dentro del conjunto de todos los posibles.
- b) Distribución aleatoria: cada cajón tendrá un número similar de valores asignados. La función de asignación parecerá aleatoria no presentando relación aparente entre los valores de la clave de búsqueda y alguna característica exterior visible (orden alfabético, longitud de la clave, etc.).

Un buen diseño es aquel en el que el tiempo consumido en una búsqueda es una constante (pequeña), e independiente del número de claves de búsqueda. Por el contrario, un mal diseño es aquel en el que el tiempo es proporcional al número de claves.

Las diferentes técnicas que hay son:

- Conversiones: consiste en convertir la clave en un número entero (o flotante utilizando la mantisa) para obtener a partir de ese valor el número de cubeta.
- Módulos: consiste en calcular la suma de las representaciones binarias de los caracteres de una clave y devolver el resto de dividir ese valor entre el número de cajones. Es la técnica más utilizada.
- Plegado: consiste en escoger y combinar los bits de la clave mediante alguna operación booleana como puede ser el XOR (OR exclusivo).

Cajón 0

|           |  |
|-----------|--|
| 10-1-1999 |  |
| 10-1-2002 |  |

Cajón 1

|           |  |
|-----------|--|
| 14-6-2000 |  |
|           |  |

Cajón 2

|  |  |
|--|--|
|  |  |
|  |  |

Cajón 3

|  |  |
|--|--|
|  |  |
|  |  |

Cajón 4

|           |  |
|-----------|--|
| 13-3-2001 |  |
| 13-3-2001 |  |

Cajón 5

|           |  |
|-----------|--|
| 20-6-2001 |  |
| 12-6-2000 |  |

|           |  |
|-----------|--|
| 24-4-2000 |  |
|           |  |

|         |       |           |
|---------|-------|-----------|
| Blanes  | Ana   | 10-1-1999 |
| Caño    | Juan  | 13-3-2001 |
| Caño    | Luis  | 20-6-2001 |
| Fuertes | José  | 10-1-2002 |
| López   | Iván  | 12-6-2000 |
| López   | Inés  | 24-4-2000 |
| López   | Eva   | 13-3-2001 |
| Montes  | Diana | 13-3-1999 |
| Mures   | Rosa  | 14-6-2000 |

|           |  |
|-----------|--|
| 13-3-1999 |  |
|           |  |

La función de asignación utilizada es la suma de los dígitos de la fecha. Después se divide entre 5 y el resto indica el cajón donde se guarda ese índice. Por ejemplo, para el registro de Ana Blanes, la suma de los dígitos de la fecha 10-1-1999 es  $1+0+1+1+9+9+9 = 30$ . Al dividirlo entre 5 da 0, con lo que se almacena en el cajón 0.

### 3.1.3. Desbordamiento de cajones.

Las principales causas del desbordamiento son:

- Número insuficiente de cajones: si se conoce el número total de registros que serán almacenados, debe cumplirse que:

$$n_c > \frac{n_{CB}}{f_{CB}} \Rightarrow \begin{cases} n_c = \text{nº de cajones} \\ n_{CB} = \text{nº total de claves de búsqueda que serán almacenadas} \\ f_{CB} = \text{nº de claves de búsqueda por cajón} \end{cases}$$

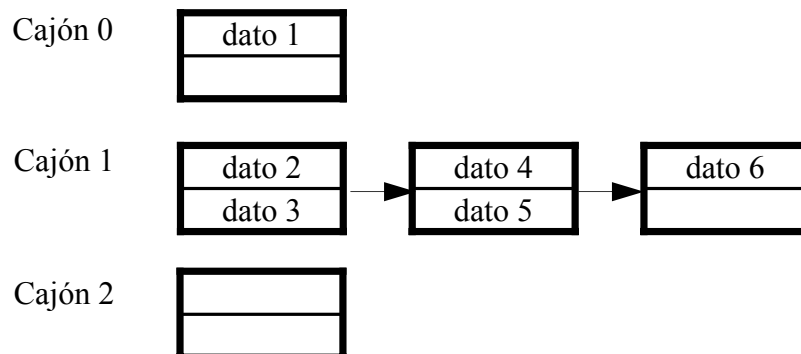
b) Atasco: unos cajones tienen asignadas más claves de búsqueda que otros. Se denomina **factor de relleno** al número de índices por página de memoria. Puede ser del 100% o menor para que si se añade alguno no haya que modificar la estructura.

- Las causas:
  - Varios registros tienen la misma clave de búsqueda.
  - La función de asociación distribuye las claves de búsqueda de forma irregular.
- Las posibles soluciones:
  - a) Reserva de espacio: reservar, aproximadamente, el 20% más de cajones necesarios.

$$n_c > \frac{n_{CB}}{f_{CB}} (1 + d); \text{ con } d \approx 0,2$$

- b) Cajones de desbordamiento: se pueden utilizar de dos formas:
- b.1) Asociación cerrada: si no cabe en un cajón, se crea un nuevo cajón llamado cajón de desbordamiento. Esta asociación utiliza cadenas de desbordamiento, con lo cual, un dato que corresponda a un cajón puede estar en el cajón o en cualquier lugar de la cadena de desbordamiento (lista enlazada de los cajones de desbordamiento creados).
  - b.2) Asociación abierta: ésta utiliza lugares libres en algún otro cajón del conjunto inicial, pero acarrea un borrado muy costoso.

Cuando el número de cajones es fijo, el rendimiento disminuye a medida que la tabla crece, debido a la aparición de cada vez más cadenas de desbordamiento.



### 3.2. ASOCIACIÓN DINÁMICA.

Son técnicas que permiten modificar la función de asociación a medida que aumenta o disminuye el tamaño de las tablas. Se obtienen mejores resultados cuando hay muchas inserciones y borrados.

#### 3.2.1. Asociación extensible.

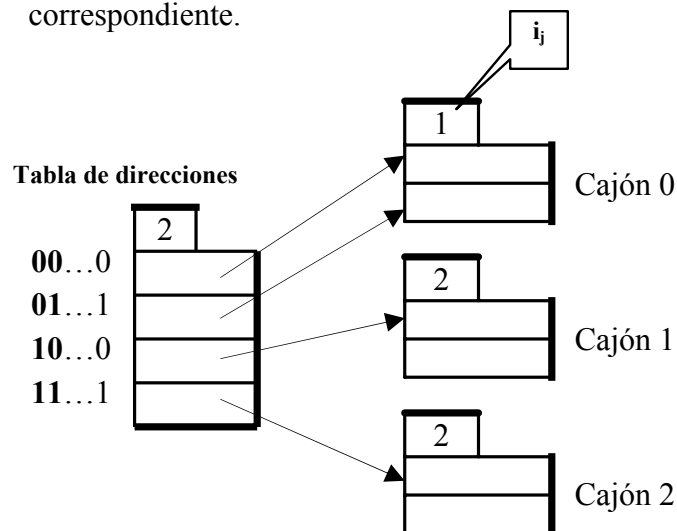
La función *hash* devuelve valores de “b” bits (normalmente  $b = 32$ ), pero no hay un cajón creado para cada posible valor de la función.

$$\{b = 32 \Rightarrow 2^{32} \approx 4 \text{ billones}\}$$

Se utilizan solamente “i” bits en cada momento ( $0 \leq i \leq b$ ). El valor inicial de i es 0, y aumentará o disminuirá con el tamaño de la base de datos.

Para determinar el cajón correspondiente a la clave  $K_i$ :

1. Calcular  $h(K_i) = x$ .
2. Tomar los “i” bits de mayor peso de x.
3. Usar los i bits como desplazamiento dentro de una tabla de direcciones de cajones y seguir el puntero hasta el cajón correspondiente.



En cada dirección hay un puntero al cajón correspondiente.

#### ▲ Inserción de una clave de búsqueda.

Si hay sitio en el cajón adecuado como resultado de calcular  $h(K_i)$ , insertar la clave de búsqueda y su puntero en ese cajón. Si el cajón está lleno, se divide en dos y se distribuyen la clave nueva y todas las que había en el cajón antiguo. Pueden darse dos situaciones diferentes:

a)  $i = i_j$

Se duplica la tabla de direcciones de cajones sustituyendo cada entrada por otras dos, teniendo cada una el mismo puntero que la original.

Se aumenta  $i$  en una unidad.

- o En este momento, dos entradas apuntan al cajón  $j$ . Se crea otro cajón,  $z$ , y se hace que la segunda entrada apunte al cajón  $z$ .
- o Se ponen  $i_j$  e  $i_z$  a valor  $i$ .
- o Se calcula la función *hash* para cada registro del cajón  $j$  y se cogen los  $i$  primeros bits para llevarlos a su cajón correspondiente.
- o Se intenta colocar la nueva entrada. Si no fuera posible, se siguen dividiendo cajones.

Si todas las claves del cajón tienen el mismo valor de búsqueda no servirá ningún número de divisiones siendo necesario utilizar cajones de desbordamiento.

b)  $i > i_j$

- o Se crea un nuevo cajón,  $z$ , y se ponen  $i_j$  e  $i_z$  al valor  $(i_j + 1)$ .
- o De la tabla de direcciones la primera mitad de las entradas que apuntaban a  $j$  se dejan como están y la otra mitad se ponen apuntando a  $z$ .
- o Se vuelve a intentar la inserción. Si volviera a fallar se aplica nuevamente uno de los dos casos.

### ♣ Borrado de una clave de búsqueda.

Si como resultado de eliminar un registro no quedan más con el mismo valor de clave de búsqueda, entonces se elimina el valor de esta clave del cajón correspondiente.

Cuando un cajón  $j$  queda vacío, se elimina dicho cajón. Pueden darse dos situaciones diferentes:

a)  $i_j < i$ .

1. Se asignan los punteros asociados al cajón  $j$  a uno de los cajones adyacentes,  $k$  ( $k = j \pm 1$ ) que verifique las siguientes condiciones:
  - $i_k = i_j$
  - Las entradas de la tabla de direcciones de cajones que apuntan al cajón  $k$  tienen los  $i_j - 1$  bits de mayor peso iguales que los de las entradas que apuntaban al cajón  $j$ .
2. Se modifica  $i_k = i_k - 1$ .

b)  $i_j = i$ .

1. Se actúa de igual manera que en la situación a).

2. Si no existe ningún cajón,  $m$ , tal que  $i_m = i$ , entonces:
  - $i = i - 1$ ;
  - Se reduce la tabla de direcciones de cajones a la mitad de tamaño.
  - Se asignan las entradas de la tabla de direcciones de cajones de forma que a cada cajón  $n$  se asocien  $2^{i-in}$  punteros.

## 4. RECOMENDACIONES DE USO.

- **Índices ordenados.**

Preferibles cuando la mayoría de las consultas especifican un rango de valores.

```
SELECT      C1, C2, C3
FROM        tabla1
WHERE       C1 > valor1 AND C1 < valor2
```

Se busca *valor1* utilizando el índice y luego se recorre secuencialmente el archivo (si es clave primaria) hasta encontrar *valor2*.

- **Índices asociativos.**

Preferibles cuando la mayoría de las consultas especifican un valor y no un rango.

```
SELECT      C1, C2, C3
FROM        tabla1
WHERE       C1 = valor1
```

Con índices ordenados el tiempo de búsqueda depende del tamaño de la tabla. Con funciones *hash* el tiempo de búsqueda no depende del tamaño.

Normalmente se usa la indexación ordenada a menos que se sepa de antemano que las consultas sobre un rango de valores van a ser poco frecuentes. Dentro de la indexación ordenada se prefieren los árboles B+ cuando hay muchas inserciones y borrados y se desea mantener la eficiencia.

### 4.1. ALGORITMOS PARA INSERTAR Y BORRAR CLAVES EN ÁRBOLES.

#### 4.1.1. Insertar:

si  $i \neq i_j$  entonces

```
incrementar  $i_j$  en 1;
duplicar el cajón;
repartir punteros;
reasignar claves;
```

si  $i = i_j$  entonces

```
incrementar  $i$  en 1;
duplicar la tabla de direcciones;
incrementar  $i_j$  en 1;
duplicar el cajón;
repartir punteros;
```

reasignar claves;

#### 4.1.2. Borrar:

si  $i \neq i_j$  entonces

eliminar cajón vacío;

asignar punteros a adyacentes  $[k]$  con

- $i_k = i_j$
- $i_j - 1$  bits de mayor peso comunes

hacer  $i_k = i_k - 1$

si  $i = i_j$  entonces

eliminar cajón vacío;

asignar punteros al adyacentes  $[k]$  con

- $i_k = i_j$
- $i_j - 1$  bits de mayor peso comunes

si no hay más cajones con  $i_k = i$

- se reduce  $i$  ( $i = i - 1$ )
- se reorganiza la tabla





# TEMA 7. TRANSACCIONES

## 1. INTRODUCCIÓN.

Se utiliza el término transacción porque los sistemas ocasionalmente sufren caídas, fallos a nivel de hardware, etc., y se necesita un mecanismo de recuperación en una base de datos. En unas, es el usuario el encargado de hacer copias de seguridad; en otras, se hace automáticamente cada cierto periodo de tiempo.

### 1.1. RECUPERACIÓN.

La recuperación en un sistema de base de datos significa restablecer la misma a un estado consistente. Esto es que satisfaga todas las restricciones de integridad conocidas.

La recuperación se basa en la redundancia a nivel físico; es decir, una base de datos es recuperable si cualquier parte de la información que contiene puede ser reconstruida a partir de otra información guardada redundantemente en algún otro lugar del sistema.

### 1.2. OBSERVACIONES.

Algunas ideas sobre la recuperación y el procesamiento de transacciones en general:

- Independientes del modelo de datos.
- Para bases de datos “grandes”, es normal que haya un acceso compartido por diferentes aplicaciones y que sean multiusuario.

### 1.2.1. Diferencia entre consistente y correcto.

Un estado consistente puede ser incorrecto si no refleja con precisión la situación tal y como es en el mundo real. Con un mecanismo de recuperación, sólo se puede garantizar que la base de datos está en un estado consistente, no correcto.

Ejemplo: Hay una restricción (check) para que los sueldos de los trabajadores de una empresa estén entre 500 € y 3000 €. Cualquier cantidad que se ponga en este campo y esté comprendida entre ambas cifras será consistente, pero no será correcta si, por ejemplo, el director tiene el sueldo más bajo posible (500 €).

## 2. TRANSACCIONES.

### 2.1. CONCEPTO.

Es una unidad de trabajo lógica, formada generalmente por una secuencia de operaciones de la bases de datos que transforman un estado consistente en otro estado consistente sin que sea necesario conservar la consistencia en todos los puntos intermedios.

### 2.2. EJEMPLO.

```
BEGIN TRANSACTION
{Se realiza una transferencia de 1000 € de la cuenta 111 a la cuenta 888.
Los saldos antiguos se han leído antes e introducido en las variables
@SaldoCuentaxxx}
UPDATE      cuentas
SET          saldo = @SaldoCuenta111 - 1000
WHERE       CuentaID = 111
IF @@Error THEN GOTO Deshacer END {fin del IF}
UPDATE      cuentas
SET          saldo = @SaldoCuenta888 + 1000
WHERE       CuentaID = 888
IF @@Error THEN GOTO Deshacer END {fin del IF}
COMMIT
GOTO        Acabar
Deshacer:   ROLLBACK      {Esto son etiquetas}
Acabar:    RETURN
```

La variable del sistema @@ERROR recoge el último error que se ha producido, devolviendo 0 si la última instrucción Transact-SQL se ejecutó con éxito; si la instrucción causó un error, @@ERROR guarda el número de error. El valor de @@ERROR cambia al finalizar cada instrucción Transact-SQL.

### 2.3. INFORMACIÓN SOBRE GOTO Y RETURN.

**GOTO:** altera el flujo de ejecución y lo dirige a una etiqueta. Las instrucciones Transact-SQL que siguen a una instrucción GOTO se pasan por alto y el procesamiento continúa en el punto que marca la etiqueta. Las instrucciones GOTO y las etiquetas se pueden utilizar en cualquier punto de un procedimiento, lote o bloque de instrucciones. Las instrucciones GOTO se pueden anidar.

Sintaxis:

Definición de la etiqueta:

label:

Alteración de la ejecución:

GOTO label

**RETURN:** sale incondicionalmente de una consulta o procedimiento. RETURN es inmediato y completo, y se puede utilizar en cualquier punto para salir de un procedimiento, lote o bloque de instrucciones. Las instrucciones que siguen a RETURN no se ejecutan.

## 2.4. OBSERVACIONES.

Una única operación atómica, que consiste en la transferencia de una cantidad de dinero de una cuenta a otra, requiere dos actualizaciones de la base de datos, pero ésta no controla la consistencia entre las dos cuentas. Hay un momento en que se añade dinero en una y no se quita de la otra, con lo que pasa por estados inconsistentes. La condición de consistencia es que la suma de los saldos de las dos cuentas ha de permanecer constante. Por lo tanto, la transacción o se ejecuta o se cancela totalmente, como si nunca hubiera sido ejecutada. Así, una secuencia de operaciones aparece como atómica, externamente.

Entonces, ¿cómo garantizar que las dos actualizaciones se realicen? Pues mediante un sistema que soporte la administración de transacciones; este sistema garantiza que si la transacción realiza alguna actualización y se produce un fallo del sistema, esa actualización será deshecha. Para garantizar esa “atomicidad”, se utilizan las siguientes funciones:

**COMMIT:** que marca el final de una transacción satisfactoria. Si éste se ejecuta, entonces todas las actualizaciones efectuadas por esa unidad de trabajo pueden ser “confirmadas”. Se guardan los cambios temporales que se habían hecho y se confirman.

**ROLLBACK:** marca el final de una transacción NO satisfactoria. Si se ejecuta, todas las actualizaciones realizadas hasta ese momento por la unidad de trabajo lógica deben ser “revertidas”. Se eliminan las operaciones no confirmadas.

Para deshacer una actualización, hay que recurrir a una bitácora, o diario mantenido por el sistema, que guarda las imágenes anterior y posterior del objeto actualizado, es decir apunta las operaciones intermedias antes de volcarlo a la base de datos.

## 3. RECUPERACIÓN DE TRANSACCIONES.

### 3.1. ESTRUCTURA.

Inicio transacción: BEGIN TRANSACTION

Fin transacción:

COMMIT: establece un punto de confirmación (o punto de sincronización). Corresponde al final de una unidad de trabajo

lógica. Este es el punto en el que la base de datos está en un estado consistente.

ROLLBACK: regresa la base de datos al estado anterior al *begin transaction*. Es decir, se regresa al punto de confirmación anterior, devolviendo la base de datos a un estado consistente.

### 3.2. CUANDO SE ESTABLECE UN PUNTO DE CONFIRMACIÓN.

- I. Se confirman todas las actualizaciones hechas por el programa en ejecución desde el punto de confirmación anterior, con lo que se vuelven permanentes.
- II. Se liberan todos los bloqueos de tuplas y se pierde el posicionamiento de la base de datos (se aplica tanto al *Commit* como al *RollBack*).

### 3.3. UNIDAD DE RECUPERACIÓN.

- ❑ Una vez que una transacción ha sido confirmada no puede deshacerse.
  - ❑ El posicionamiento de la base de datos se refiere a que en cualquier momento un programa en ejecución tendrá una direccionalidad hacia ciertas tuplas (puede ser mediante cursores).
  - ❑ Un *commit* o *rollback* finalizan una transacción pero no el programa en el que se encuentran, que suele estar formado por una secuencia de transacciones. Una terminación planeada es un *commit* o *rollback* explícito (definido por el programador).
  - ❑ El sistema emitirá una instrucción ROLLBACK implícita para cualquier transacción que falle antes de llegar a su terminación planeada.
  - ❑ Al confirmar una transacción, el sistema garantiza que sus actualizaciones quedarán grabadas permanentemente aunque el sistema falle en el instante siguiente. El procedimiento de reinicio grabará las actualizaciones aunque el sistema haya fallado y no se hayan escrito después del COMMIT.
- **Regla de escritura anticipada de la bitácora:** se escribe antes de terminar el procesamiento del COMMIT.

### 3.4. PROPIEDADES PACA (ACID).

- 1) *Persistencia*: una vez confirmadas permanecen en la base de datos aun cuando haya una caída posterior del sistema.
- 2) *Atomicidad*: o todo o nada.
- 3) *Consistencia*: transforman un estado consistente en otro también consistente (aunque no lo sea en puntos intermedios).
- 4) *Aislamiento*: aisladas entre sí. En general hay muchas ejecutándose concurrentemente pero las actualizaciones de una transacción están ocultas a las demás hasta que sea confirmada.

## 4. RECUPERACIÓN DEL SISTEMA.

### 4.1. FALLOS DURANTE LA EJECUCIÓN.

- a) *Fallos locales*: por ejemplo los desbordamientos, violación de restricciones.
- b) *Fallos globales*.
  - b.1) *Fallos del sistema*. (Ej. suministro eléctrico). Afecta a todas pero no dañan físicamente la base de datos (se denomina caída blanda). Con el registro de transacciones se vuelve a un estado consistente.
  - b.2) *Fallos del medio*. (Ej. roce de las cabezas con el disco). Causan daño a la base de datos (entera o parte), afectando al menos a las transacciones que están utilizando esa parte

Cuando se produce un fallo del sistema, se pierde el contenido de la memoria principal, por lo que es necesario deshacer las transacciones en progreso no completadas y rehacer las completadas pero no transferidas de los buffers de la base de datos hacia la bases de datos física.

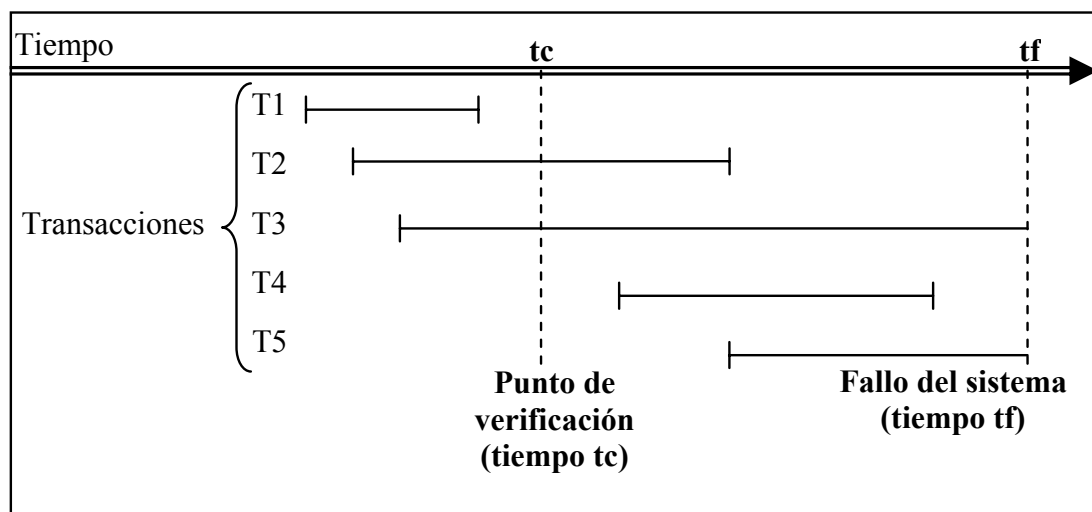
### 4.2. PUNTO DE VERIFICACIÓN.

La pregunta es: ¿qué transacciones deshacer y cuáles rehacer?

A determinados intervalos prescritos, generalmente tras cierta cantidad de entradas en la bitácora, el sistema automáticamente realiza un punto de verificación:

- a) Escribe el contenido de los buffers hacia la base de datos física.
- b) Escribe físicamente un registro de punto de verificación en la bitácora física.

Se produce, por tanto, una lista de todas las transacciones en progreso en el momento de escribir el registro.



En el reinicio: deben deshacerse T3 y T5, y deben rehacerse T2 y T4. Como en el punto de verificación T1 está confirmada, se graba en ese instante **tc**.

### 4.3. PROCEDIMIENTO EN EL REINICIO.

Consiste en dos pasos:

#### 4.3.1. Identificación.

Se especifican qué transacciones hay que deshacer y cuáles hay que rehacer.

1. Parte del punto de verificación y comienza con dos listas de transacciones: DESHACER y REHACER. REHACER inicialmente está vacía. DESHACER contiene todas las transacciones apuntadas en el último registro de punto de verificación.
2. Busca hacia delante en la bitácora a partir de dicho registro.
3. Si se encuentra en la bitácora una entrada *begin transaction* para la transacción T, añade T a DESHACER.
4. Si se encuentra una entrada *commit* para T, mueve T de la lista DESHACER a la lista REHACER.
5. Cuando llega al final de la bitácora, la lista DESHACER contiene las transacciones de tipo T3 y T5 y la lista REHACER las transacciones de tipo T2 y T4.

#### 4.3.2. Procesado.

El sistema trabaja de la siguiente forma:

- a) Hacia atrás en la bitácora, deshaciendo las transacciones que están en la lista DESHACER.
- b) Y luego hacia delante, volviendo a realizar las transacciones de la lista REHACER.

A la acción de devolver la base de datos a un estado consistente deshaciendo/rehaciendo el trabajo se le llama **recuperación hacia atrás/adelante**.

# TEMA 8. CONCURRENCIA

## 1. INTRODUCCIÓN.

¿Qué pasa cuando un sistema gestor de bases de datos permite que muchas transacciones accedan a una misma base de datos a la vez? Pueden producirse problemas cuando varias transacciones intentan acceder a los mismos datos. Para evitar estos problemas, utilizan mecanismos de control de concurrencia para asegurar que las transacciones concurrentes no interfieran entre sí.

Como los mecanismos de control de concurrencia se basan en las transacciones, se debe tener muy clara cuál es la diferencia entre una base de datos en estado consistente (aquella que es congruente con las restricciones definidas en la misma) y una base de datos en estado correcto. Un estado consistente puede ser incorrecto si no refleja con precisión el verdadero estado de las cosas en el mundo real. Ej.: Hay una restricción (check) para que los sueldos de los trabajadores de una empresa estén entre 500 € y 3000 €. Cualquier cantidad que se incluya en este campo y esté comprendida entre ambas cifras será consistente, pero no será correcta si, por ejemplo, el director tiene el sueldo más bajo posible (500 €).

## 2. PROBLEMAS DE LA CONCURRENCIA.

Situación: cuando las transacciones son concurrentes, las cosas pueden “salir mal”, es decir, aunque una transacción sea correcta por sí misma, puede producir una respuesta incorrecta si alguna otra transacción interfiere con ella en alguna forma.

Lo que produce el resultado incorrecto es el *intercalado* sin control entre las operaciones de las dos transacciones correctas.

Hay tres posibles problemas a destacar en los estados de concurrencia:

1. El problema de la *actualización perdida*.

2. El problema de la *dependencia no confirmada*.
3. El problema del *análisis inconsistente*.

### 2.1. LA ACTUALIZACIÓN PERDIDA.

Se realiza una actualización de un objeto cualesquiera de la base de datos y se pierde dicha actualización.

#### Situación:

A recupera una tupla T en el tiempo t1  
 B recupera la misma tupla T en el tiempo t2  
 A actualiza T en t3  
 B actualiza T en t4

| Transacción A | Tiempo | Transacción B |
|---------------|--------|---------------|
| ---           |        | ---           |
| ---           |        | ---           |
| Recuperar T   | t1     | ---           |
| ---           |        | ---           |
| ---           | t2     | Recuperar T   |
| ---           |        | ---           |
| Actualizar T  | t3     | ---           |
| ---           |        | ---           |
| ---           | t4     | Actualizar T  |
| ---           |        | ---           |

#### Conclusión:

La actualización de la transacción A se pierde en el tiempo t4, al sobrescribirla la transacción B en el momento t4.

### 2.2. LA DEPENDENCIA NO CONFIRMADA.

Este tipo de problema se presenta al permitir que una transacción recupere (o aún peor, actualice) una tupla que ha sido actualizada por otra transacción, pero que aún no ha sido confirmada por esa misma transacción.

Puede que no se confirme nunca y que se deshaga con un ROLLBACK. Por ejemplo, puede que se deshaga la transacción "B" al producirse una caída del sistema. En el caso de que "A" termine antes de caerse el sistema no será posible deshacer "A".



| Transacción A        | Tiempo | Transacción B |
|----------------------|--------|---------------|
| ---                  |        | ---           |
| ---                  | t1     | Actualizar T  |
| ---                  |        | ---           |
| Recuperar T (Act. T) | t2     | ---           |
| ---                  |        | ---           |
| ---                  | t3     | ROLLBACK      |
| ---                  |        | ---           |

En la figura anterior se puede observar que si A realiza una actualización en vez de una recuperación en t2, la perderá en t3 ya que el ROLLBACK hace que T quede al valor que tenía en t1, es decir, se vuelve al último estado consistente de la base de datos.

### 2.3. EL ANÁLISIS INCONSISTENTE.

Se produce cuando una transacción ve un estado inconsistente.

#### Situación.

La transacción A calcula la suma total de los saldos de las cuentas CC1, CC2 y CC3.

La transacción B realiza un movimiento de capital desde la cuenta CC3 a la cuenta CC1, con lo que se realizan dos operaciones de recuperación de datos y dos de actualización de los mismos.

| CC1        | CC2        | CC3        |
|------------|------------|------------|
| Saldo = 40 | Saldo = 50 | Saldo = 30 |

| Transacción A                              | Tiempo | Transacción B               |
|--|--------|-----------------------------|
| ---  |        | ---                         |
| Recuperar CC1:<br>suma = 40                | t1     | ---                         |
| ---  |        | ---                         |
| Recuperar CC2:<br>suma = 90                | t2     | ---                         |
| ---  | t3     | Recuperar CC3               |
| ---  | t4     | Actualizar CC3:<br>30 -> 20 |
| ---  | t5     | Recuperar CC1               |
| ---  | t6     | Actualizar CC1:<br>40 -> 50 |
| ---  | t7     | COMMIT                      |
| Recuperar CC3:<br>suma = 110<br>(y no 120) | t8     | ---                         |

**Conclusión.**

“A” ve un estado inconsistente, dado que no obtiene el saldo inicial de la cuenta CC3 sino que recupera el saldo de la cuenta ya actualizado por la transacción B y, por lo tanto, ha realizado un análisis inconsistente.

“A” no es dependiente de un cambio no confirmado (B realiza COMMIT antes de que A lea CC3). La diferencia con el problema anterior es que la transacción está confirmada, pero el estado que ha visto la otra transacción no es consistente.

**3. BLOQUEOS.**

Los bloqueos permiten resolver problemas de concurrencia.

Cuando una transacción deba asegurarse que un objeto en el que está interesada –por lo general una tupla- no cambiará mientras lo está usando, adquiere un bloqueo sobre ese objeto. Un bloqueo inhibe todas las demás transacciones en un objeto impidiendo que otras lo cambien.

**3.1. FUNCIONAMIENTO DE LOS BLOQUEOS.****3.1.1. Tipos de bloqueos.**

Se pueden definir dos tipos distintos de bloqueos:

1. Bloqueos exclusivos (X) (o bloqueo de escritura). Si la transacción A pone un bloqueo exclusivo (X) sobre la tupla T, entonces ninguna otra transacción podrá realizar ningún tipo de bloqueo sobre T.
2. Bloqueos compartidos (S) (o bloqueo de lectura). Si la transacción A pone un bloqueo compartido (S) sobre la tupla T, entonces:
  - Se rechazará una petición de cualquier otra transacción B para un bloqueo X sobre T.
  - Se otorgará una petición de cualquier otra transacción B para un bloqueo S sobre T.

**3.1.2. Matriz de compatibilidad de tipos de bloqueo.**

La compatibilidad entre estos bloqueos queda representada por la siguiente matriz:

|               |   | Nueva solicitud |    |    |
|---------------|---|-----------------|----|----|
|               |   | X               | S  | -  |
| Estado actual | X | No              | No | Sí |
|               | S | No              | Sí | Sí |
|               | - | Sí              | Sí | Sí |

**X** representa un bloqueo exclusivo o de escritura.

**S** representa un bloqueo compartido o de lectura.

- representa la ausencia de bloqueo.

**No:** indica que van a existir conflictos, es decir, la petición de bloqueo no va a ser satisfecha con lo que la transacción que la genere entrará en un estado de espera.

**Si:** indica que no existe ningún tipo de conflicto.

### 3.2. PROTOCOLO DE ACCESO A DATOS.

Atendiendo a los bloqueos se puede obtener un protocolo que indique la manera correcta de acceder a los datos para evitar el problema de la concurrencia:

1. Una transacción que desea recuperar una tupla primero debe adquirir un bloqueo S sobre esa tupla.
2. Una transacción que desea actualizar una tupla primero debe adquirir un bloqueo X sobre esa tupla. Si ya tiene un S debe promoverlo a X (para actualizar se necesita tener un bloqueo exclusivo).

Las peticiones de bloqueo son implícitas, es decir, una petición de recuperación de tupla es una petición implícita de un bloqueo S y una petición de actualización de tupla es una petición implícita de un bloqueo X sobre la tupla relevante.

Se debe tener en cuenta que las operaciones de actualización también incluyen los INSERT y DELETE y, por supuesto, los UPDATE.

3. Cuando se rechaza una petición de bloqueo de una transacción B porque entra en conflicto con un bloqueo que ya tiene la transacción A, la transacción B pasa a un estado de espera hasta que se libere el bloqueo. Hay que evitar la espera indefinida mediante alguna técnica. Un método posible puede ser una cola FIFO.

4. Los bloqueos se mantienen:

X: hasta el final de la transacción (COMMIT o ROLLBACK).

S: normalmente también hasta el final de la transacción.

## 4. NUEVAMENTE LOS TRES PROBLEMAS.

### 4.1. LA ACTUALIZACIÓN PERDIDA.

#### Situación.

En el momento t1 la transacción A recupera la tupla T con lo que adquiere un bloqueo compartido de T. En el momento t2 la transacción B recupera la tupla T con lo que adquiere un bloqueo compartido de T (entre dos bloqueos de lectura no se produce conflicto alguno). En el momento t3 la transacción A intenta actualizar la tupla T pero su petición de bloqueo de escritura sobre la tupla T no puede ser satisfecha por lo que la transacción A entra en un estado de espera. En el momento t4 la transacción B intenta realizar una actualización en la tupla T pero su petición de bloqueo de escritura sobre la tupla T no

puede ser satisfecha por lo que la transacción B también entra en un estado de espera.

| Transacción A                                | Tiempo | Transacción B                                |
|--|--------|--|
| ---  |        | ---  |
| ---  |        | ---  |
| Recuperar T(adquiere<br>bloqueo S sobre T)   | t1     | ---  |
| ---  |        | ---  |
| ---  | t2     | Recuperar T (adquiere<br>bloqueo S sobre T)  |
| ---  |        | ---  |
| Actualizar T (solicita<br>bloqueo X sobre T) | t3     | ---  |
| espera                                       |        | ---  |
| espera                                       | t4     | Actualizar T (solicita<br>bloqueo X sobre T) |
| espera                                       |        | espera                                       |
| espera                                       |        | espera                                       |

El sistema de bloqueos resuelve el problema de la actualización perdida, pero lo convierte en un nuevo problema: **el bloqueo mortal**.

#### 4.2. LA DEPENDENCIA NO CONFIRMADA.

**CASO 1:** Recuperación de datos.

| Transacción A   | Tiempo | Transacción B                                   |
|---|--------|---|
| ---   |        | ---   |
| ---   | t1     | Actualizar T<br>(adquiere bloqueo X sobre T)    |
| ---   |        | ---   |
| Recuperar T<br>(solicita bloqueo S sobre T)                 | t2     | ---   |
| espera  |        | ---   |
| espera  | t3     | COMMIT / ROLLBACK<br>(libera bloqueo X sobre T) |
| espera  |        |   |
| <b>Reanuda:</b> Recuperar T<br>(adquiere bloqueo S sobre T) | t4     |   |
| ---   |        |   |

##### Situación.

En t1 la transacción B realiza una actualización de la tupla T por ello establece un bloqueo exclusivo sobre la misma. En t2 la transacción A intenta recuperar la tupla T, pero el bloqueo de lectura requerido no se

le puede proporcionar a causa del conflicto con el bloqueo exclusivo de B.

A permanece en espera hasta que B llega a su término (libera bloqueo). Cuando A ve un valor confirmado continúa.

### Conclusión.

A no puede ver un cambio no confirmado en t2.

### CASO 2: Actualización de datos.

| Transacción A  | Tiempo | Transacción B                                   |
|--|--------|---|
| ---  |        | ---   |
| ---  | t1     | Actualizar T<br>(adquiere bloqueo X sobre T)    |
| Actualizar T<br>(solicita bloqueo X sobre T)                 | t2     | ---   |
| espera   |        | ---   |
| espera   | t3     | COMMIT / ROLLBACK<br>(libera bloqueo X sobre T) |
| espera   |        |   |
| <b>Reanuda:</b> Actualizar T<br>(adquiere bloqueo X sobre T) | t4     |   |
| ---  |        |   |

### Situación.

En t1 la transacción B realiza una actualización de la tupla T por ello establece un bloqueo exclusivo sobre la misma. En t2 la transacción A intenta actualizar la tupla T, pero el bloqueo de escritura requerido no se le puede proporcionar a causa del conflicto con el bloqueo exclusivo de B.

A permanece en espera hasta que B llega a su término (libera bloqueo). Cuando A ve un valor confirmado continúa.

### Conclusión.

A no puede actualizar un cambio no confirmado en t2.

### 4.3. EL ANÁLISIS INCONSISTENTE.

| CC1<br>Saldo =40  | CC2<br>Saldo = 50 | CC3<br>Saldo = 30   |
|---|-------------------|---|
| Transacción A   | Tiempo            | Transacción B   |
| ---   |                   | ---   |
| Recuperar CC1: (adquiere<br>bloqueo S sobre CC1)<br>suma = 40 | t1                | ---   |
| ---   |                   | ---   |
| Recuperar CC2: (adquiere<br>bloqueo S sobre CC2)<br>suma = 90 | t2                | ---   |
| ---   | t3                | Recuperar CC3<br>(adquiere bloqueo S sobre CC3)               |
| ---   | t4                | Actualizar CC3:<br>(adquiere bloqueo X sobre CC3)<br>30 -> 20 |
| ---   | t5                | Recuperar CC1<br>(adquiere bloqueo S sobre CC1)               |
| ---   | t6                | Actualizar CC1:<br>(solicita bloqueo X sobre CC1)<br>40 -> 50 |
| ---   |                   | espera  |
| Recuperar CC3: (solicita<br>bloqueo S sobre CC3)<br>espera    | t7                | espera  |
|   |                   | espera  |

Nuevamente el bloqueo resuelve el problema del análisis inconsistente, aunque forzando un bloqueo mortal.

### 4.4. BLOQUEO MORTAL.

Un estado de bloqueo mortal se producirá cuando dos o más transacciones se encuentran en estados simultáneos de espera, esperando cada una de ellas que alguna de las demás libere un bloqueo para poder continuar. Son posibles bloqueos mortales que involucren a más de dos transacciones (tres y cuatro) pero no son habituales.

| Transacción A        | Tiempo | Transacción B        |
|----------------------|--------|----------------------|
| ---                  |        | ---                  |
| Bloqueo r1 exclusivo | t1     | ---                  |
| ---                  | t2     | Bloqueo r2 exclusivo |
| ---                  |        | ---                  |
| Bloqueo r2 exclusivo | t3     | ---                  |
| espera               |        | ---                  |
| espera               | t4     | Bloqueo r1 exclusivo |
| espera               |        | espera               |

La tabla anterior puede servir como versión general de un bloqueo mortal. En esta representación **r** son los recursos (cualquier objeto bloqueable) y **“Bloqueo ...exclusivo”** representa cualquier operación que adquiera bloqueos exclusivos (explícita o implícita).

El sistema, ante los bloqueos mortales, debe detectarlos y romperlos.

**Detección:** encontrar ciclos en el grafo de espera. El grafo de espera es el grafo que representa “quién está esperando a quién”.

**Rotura:** seleccionar una víctima y deshacerla (ROLLBACK), liberando así sus bloqueos. En el caso en el que la víctima ha fallado y ha sido deshecha sin ser culpable, algunos sistemas volverán a iniciar esta transacción desde el principio.

No todos los sistemas detectan bloqueos mortales sino que usan algún mecanismo de tiempo y asumen simplemente que si una transacción no ha realizado ningún trabajo durante cierto periodo de tiempo está bloqueada mortalmente. La gestión de transacciones a las que hace ROLLBACK por bloqueo exclusivo depende del sistema. Unos le dan a la transacción víctima (la que hizo el ROLLBACK) una oportunidad cuando acaba la que ganó el conflicto. Otros sistemas devolverán un código de excepción hacia la aplicación y tendrá que manejar la situación el programador, reiniciando la transacción víctima.

## 4.5. SERIABILIDAD.

### 4.5.1. Concepto.

Se considera que la ejecución de un conjunto dado de transacciones es correcta cuando es serializable, es decir, una ejecución es serializable cuando produce el mismo resultado que una ejecución en serie de las mismas transacciones.

De este modo podemos definir la seriabilidad como el criterio de corrección para la ejecución de un conjunto dado de transacciones.

La afirmación de seriabilidad es correcta porque:

1. Se toman las transacciones individuales como correctas.
2. Por lo tanto, es correcta la ejecución de una transacción a la vez en cualquier orden serial, al ser las individuales independientes entre sí.
3. Por lo tanto, una ejecución intercalada es correcta cuando equivale a una ejecución serial.

Los problemas anteriores aparecían porque la ejecución intercalada no era serializable. Los bloqueos forzaron que sí lo fuera.

#### 4.5.2. Plan.

Se puede definir como plan cualquier ejecución, intercalada o no, de un conjunto de transacciones.

Dentro de los planes existen dos subtipos:

- *Plan serial*: las transacciones se ejecutan una tras otra, sin intercalar.
- *Plan intercalado*: las transacciones se ejecutan de manera intercalada.

Dos planes son **equivalentes** cuando garantizan que producirán el mismo resultado independientemente del estado inicial de la base de datos.

Es importante destacar que un plan es correcto cuando equivale a un plan serializable.

Dos planes serializables diferentes que involucren el mismo conjunto de transacciones pueden producir resultados diferentes y, por lo tanto, dos planes intercalados diferentes que involucren a esas transacciones, también pueden producir resultados diferentes, aunque ambos sean considerados como correctos.

#### 4.5.3. Teorema del bloqueo de dos fases.

Es una generalización del protocolo de acceso a datos y se puede enunciar como:

Si todas las transacciones obedecen el “protocolo de bloqueo de dos fases”, entonces todos los planes intercalados posibles son serializables.

A partir del anterior teorema, se puede definir un protocolo de bloqueo de dos fases, en el que los pasos principales son los siguientes:

1. Antes de operar sobre un objeto, una transacción debe adquirir un bloqueo sobre ese objeto.
2. Después de liberar un bloqueo, una transacción nunca deberá adquirir ningún otro bloqueo.

Si una transacción A no es de dos fases (no obedece el protocolo de bloqueo de dos fases) siempre es posible construir alguna otra transacción B que pueda ejecutarse intercalada con A, de manera que produzca un plan general que no sea serial ni correcto.

La fase de liberar un bloqueo (reducción) suele ser o bien un COMMIT o bien un ROLLBACK.

Para reducir la disputa de recursos, y por consiguiente mejorar el rendimiento total, los sistemas reales habitualmente permiten construir transacciones que no son de dos fases, es decir, transacciones que “liberan tempranamente los bloqueos” (antes del COMMIT) y luego continúan adquiriendo más bloqueos. Para ello se aceptan los riesgos de errores.



Si se cumple el protocolo de bloqueo de dos fases entonces no hay problemas de concurrencia.



# PRÁCTICAS

MICROSOFT SQL SERVER 7.0 (MSS7)



# PRÁCTICA 1. VISIÓN GENERAL

## 1. OBJETIVOS.

Al finalizar esta práctica el alumno será capaz de:

1. Definir algunos conceptos generales relacionados con las bases de datos y con los gestores de bases de datos. Algunos de ellos son:
  - a. Escalabilidad.
  - b. Arquitectura cliente/servidor.
  - c. Duplicación.
  - d. Consola de administración de Microsoft.
  - e. Herramientas de red.
  - f. Monitor de rendimiento.
  - g. Analizador de consultas.
  - h. Plan de ejecución.
  - i. Traza.
  - j. Administrador de servicios.
  - k. Administrador corporativo.
2. Explicar qué es Microsoft SQL Server indicando alguna de sus características, qué partes tiene y para qué sirve cada parte.
3. Utilizar los libros en pantalla para buscar información.

## 2. DETALLES.

### 2.1. CUESTIONES GENERALES.

Utilizando la ayuda o los libros en pantalla, responde brevemente a las siguientes preguntas.

1. ¿En qué consiste la *escalabilidad* de una base de datos? (Ver escalabilidad del servidor).
2. ¿Por qué se caracteriza un sistema de base de datos de servidor y por qué un sistema de base de datos de escritorio?. Indique en qué categoría se puede englobar M. SQL Server 2000.
3. ¿Mediante qué (estructura de datos) se implementa físicamente una base de datos en M. SQL Server 2000? (Ver Arquitectura).
4. ¿Cuántas copias del motor de base de datos es necesario ejecutar para que varios usuarios puedan tener acceso a las bases de datos de un servidor?
5. ¿Cuántos usuarios pueden conectarse simultáneamente al mismo servidor M. SQL Server 2000 Standard o Enterprise?
6. ¿Se pueden tener varias bases de datos de usuario por cada instancia de M. SQL Server 2000?
7. ¿En qué consiste la *duplicación*?
8. ¿Cómo se implementa la *duplicación* en M. SQL Server 2000?
9. ¿Cuál es el objetivo de “optimizar el rendimiento” de una base de datos?
10. ¿Qué implica la optimización de consultas?
11. ¿Para qué se realizan copias de seguridad y en qué consiste la *restauración* de una base de datos?
12. ¿Qué información suele ser necesario proporcionarle a una aplicación para conectar con una base de datos (instancia de M. SQL Server 2000) que se encuentra en un equipo remoto? (Ver Inicios de sesión).

### 2.2. CARACTERÍSTICAS DE LA IMPLEMENTACIÓN.

Cuales son los siguientes tamaños máximos, tanto para M. SQL Server 2000 como para su antecesor M. SQL Server 7 (Ver Detalles de implementación):

1. Tamaño de una base de datos.
2. Tablas por base de datos.
3. Filas por tabla.
4. Objetos en una base de datos.
5. Número de bases de datos por instancia de SQL.
6. Índices agrupados por tabla.
7. Índices no agrupados por tabla.
8. Columnas por instrucción SELECT.
9. Tablas por instrucción SELECT.

10. Referencias a claves externas (REFERENCES) por tabla.

## **2.3. HERRAMIENTAS GRÁFICAS.**

### **2.3.1. Administrador Corporativo.**

1. ¿Qué es?
2. ¿Para qué sirve?
3. Abra el Administrador Corporativo y observe qué objetos se pueden desplegar a partir de la Raíz de la Consola y qué contenidos de esos objetos se muestran en la parte derecha.
4. ¿Qué es la MMC?
5. ¿Para qué sirve la MMC?
6. ¿Cómo se puede abrir una MMC desde la línea de comandos (símbolo del sistema)?
7. Abra una consola y mire a ver qué complementos se pueden agregar.
8. Recorra los diferentes menús y observe las opciones disponibles.

### **2.3.2. Analizador de consultas.**

1. ¿Qué es?
2. ¿Qué permite hacer?
3. ¿Qué es el isqlw?
4. ¿Se puede seleccionar la base de datos sobre la que se va a consultar?
5. Observe cómo puede conectarse a un servidor de bases de datos.
6. Recorra los diferentes menús y observe las opciones disponibles.
7. En el menú consulta observe cuales son las “Opciones de la conexión actual”, tanto en la pestaña general como en la avanzada.

### **2.3.3. Herramienta de red del cliente.**

1. ¿Para qué sirve?
2. Abra la herramienta y observe qué diferentes opciones de bibliotecas de red ofrece.

### **2.3.4. Herramienta de red del servidor.**

1. ¿Qué es?
2. Abra la herramienta y observe las configuraciones de biblioteca de red del servidor activo y las bibliotecas de red disponibles.

### **2.3.5. Monitor de sistema de Windows 2000 (y de XP).**

1. ¿Qué es?
2. ¿Qué proporciona?

3. Observe si puede abrirlo y explique razonadamente lo que observa y lo que piensa de ello.

#### **2.3.6. Analizador de SQL Server.**

1. ¿Qué es?
2. ¿Qué permite hacer?
3. ¿Qué hace con los sucesos del servidor?
4. ¿En qué actividades se utiliza?
5. ¿Qué es una traza y cómo se crea?
6. Desde el menú **Archivo** cree una traza de prueba y observe las distintas propiedades de ella en las pestañas de *general*, *sucesos*, *columnas de datos*, *filtros*.
7. Cree otra traza de prueba utilizando el asistente. Observe los problemas que puede identificar.
8. Abra diversas plantillas de trazas y observe su contenido.

#### **2.3.7. Administrador de servicios.**

1. ¿Qué es?
2. ¿Para qué se emplea?
3. ¿Sobre qué servicios se puede actuar?



# PRÁCTICA 2.

## VISIÓN GENERAL (CONTINUACIÓN)

### 1. OBJETIVOS.

Al finalizar esta práctica el alumno será capaz de:

1. Definir y/o explicar algunos conceptos generales de bases datos y otros específicos de MS SQL Server:
  - a. Bases de datos del sistema. Saber cuáles son y conocer en profundidad para qué se utiliza cada una de ellas.
  - b. Página y extensión.
  - c. Tipos de datos del sistema.
  - d. Vistas. Concepto. Detalles. Utilización.
  - e. Restricciones: concepto, tipos, características.
  - f. Desencadenadores.
  - g. Integridad: categorías y concepto.
  - h. Procedimientos almacenados.
  - i. Diagramas.
  - j. Índices.
2. Utilizar los libros en pantalla para buscar información.

## 2. DETALLES.

### 2.1. INFORMACIÓN DEL SISTEMA.

1. ¿Qué base de datos se utiliza como plantilla para todas las bases de datos creadas en un sistema? (Ver *Arquitectura de Bases de Datos*).
2. ¿Dónde se registra toda la información del sistema de SQL Server?
3. Enumere la información que se registra en la base de datos *master*.
4. ¿Dónde se almacenan todas las tablas y los procedimientos almacenados temporales?
5. ¿Dónde registra el Agente SQL Server los operadores?
6. ¿Dónde se almacenan las tablas de trabajo temporales generadas por SQL Server?
7. ¿De dónde copia la instrucción CREATE DATABASE la primera parte del contenido de una nueva base de datos?
8. ¿Dónde se registra la existencia de todas las demás bases de datos?
9. ¿Dónde se registra la ubicación de los archivos que contienen la información de inicialización de las bases de datos de usuario?
10. ¿Dónde almacena el Agente SQL Server las alertas programadas y los trabajos?
11. ¿Se mantiene un histórico en *tempdb* o se crea una base de datos limpia con cada nuevo arranque del motor de la base de datos?
12. ¿Es necesario estar pendiente del espacio que necesita *tempdb* para evitar que el espacio destinado inicialmente se agote?

### 2.2. ALMACENAMIENTO, DATOS Y OBJETOS.

1. ¿Cuál es la unidad fundamental de almacenamiento de datos en Microsoft SQL Server (Ver *Arquitectura física de la base de datos*)?
2. ¿Cuántas **páginas** por Mega tiene una base de datos y cuál es el tamaño de una página?
3. ¿Qué es una **extensión**?
4. ¿Cuántas extensiones por Mega tiene una base de datos?
5. ¿Cuántos tipos de páginas hay en SQL Server?
6. ¿Pueden las páginas de datos contener todos los tipos de datos?
7. Realice un esquema con los **tipos de datos** suministrados por el sistema (hágalo en una hoja aparte para utilizarlo como referencia en posteriores ocasiones). Ponga un ejemplo de utilización para cada uno de ellos. Agrúpelos por categorías. Intente obtener únicamente 7 categorías diferentes.
8. ¿Qué diez componentes lógicos tiene una base de datos (Ver *Componentes Lógicos de una base de datos*)?
9. ¿Qué dos componentes principales tienen las tablas de SQL Server?

10. ¿Qué es una **vista**?
11. ¿Puede una vista hacer referencia a otra vista?
12. ¿Qué es una instrucción SELECT?
13. ¿Es correcto pensar que una vista es una instrucción SELECT almacenada?
14. Enumere y explique brevemente cinco escenarios en los que se suelen utilizar las vistas.
15. ¿Qué dos tipos de **integridad** tiene que garantizar una base de datos (Ver *Arquitectura de Bases de Datos- Restricciones, ...*)?.
16. ¿Qué objetos se utilizan para mantener los dos tipos de integridad anteriores?.
17. Explique para qué se utilizan las **restricciones** y qué características tienen. Apunte estas definiciones en la hoja de referencia –junto con los tipos de datos- poniendo además los cinco tipos de restricciones que permite SQL Server con una breve definición de cada una de ellas.
18. Explique qué es un **procedimiento almacenado**.
19. Explique qué es un **desencadenador**.
20. Indique cuales son las cuatro principales utilidades de los desencadenadores.
21. Explique en qué consiste la **integridad de los datos** y enumera las cuatro categorías de integridad que hay. Realice un esquema que incluya el nombre de la categoría, una definición breve indicando en qué consiste y las opciones recomendadas para mantener dicha integridad (apúntelo en las hojas de referencia).
22. ¿Qué son y dónde se encuentran los procedimientos almacenados del sistema?
23. ¿Un procedimiento almacenado en SQL Server es similar a un procedimiento de otro lenguaje?
24. ¿Qué instrucción de Transact-SQL permite crear procedimientos almacenados?
25. ¿Se puede utilizar un procedimiento almacenado sin utilizar la palabra EXECUTE?
26. Indique si es posible conceder permisos a los usuarios para ejecutar un procedimiento almacenado, incluso si no cuentan con permiso para ejecutar directamente las instrucciones del procedimiento.

### 2.3. DISEÑADORES, DIAGRAMAS, ÍNDICES .

1. ¿Qué es el *Diseñador de bases de datos*?
2. ¿Qué utilidad tienen los **diagramas** de la base de datos y cuántos puede crearse sobre una misma base de datos?
3. ¿Qué es el *Diseñador de tablas* y qué dos partes tiene?

4. ¿Qué es un **índice**?
5. ¿El gestor de bases de datos crea automáticamente algún índice, o todos tienen que explicitarse?. En caso afirmativo indique cuáles.
6. ¿Es verdad que en una base de datos, un índice permite que el programa de la base de datos busque datos en una tabla sin necesidad de examinar toda la tabla?
7. ¿La creación de índices tiene algún inconveniente?
8. ¿Es posible, al igual que los índices normales, utilizar varios índices de texto por tabla?
9. ¿Permiten los índices mejorar el rendimiento?. ¿Por qué?

# PRÁCTICA 3.

## DISEÑO Y CREACIÓN DE BASES DE DATOS

### 1. OBJETIVOS.

Al finalizar esta práctica el alumno será capaz de:

1. Definir y/o explicar algunos conceptos generales de bases datos y otros específicos de MSQL Server:
  - a. Claves primarias y externas.
  - b. Restricciones UNIQUE.
  - c. Valores NULL.
  - d. Transacciones. Concepto. Registro de transacciones.
  - e. Bloqueos. Concepto y utilidad.
  - f. Plan de una base de datos.
  - g. Categorías de aplicaciones de bases de datos.
  - h. Normalización.
  - i. Operaciones de mantenimiento.
  - j. Particiones sin formato.
  - k. Tipos de archivos de datos.
2. Crear una base de datos utilizando el Administrador Corporativo.
3. Crear tablas y asignar el tipo de datos apropiado a cada campo.
4. Crear un diagrama de una base datos.

5. Definir claves primarias.
6. Especificar atributos NOT NULL.
7. Establecer las restricciones de integridad referencial.

## 2. DETALLES.

### 2.1. CLAVES, TRANSACCIONES Y BLOQUEOS .

1. ¿Una columna definida como clave principal puede aceptar valores NULL? (Ver *Crear y mantener bases de datos – Tablas*).
2. ¿Se puede establecer que la clave principal de una tabla esté compuesta por más de una columna?. ¿Hay alguna restricción o limitación?.
3. Explique qué es una clave externa y para qué se utiliza (apúntelo en la hoja de referencia).
4. ¿Una clave externa puede vincularse únicamente a una clave primaria?. Explíquelo.
5. ¿Una columna con la restricción UNIQUE puede aceptar valores NULL?
6. Cuántas restricciones UNIQUE y cuántas restricciones PRIMARI KEY pueden definirse en cada tabla.
7. ¿Qué es una transacción (Ver *Acceso y cambio en los datos relacionales*)?
8. ¿Es cierto que las transacciones se administran en las conexiones?
9. Enumere los tres modos de iniciar una transacción
10. ¿Las transacciones de SQL Server se inician y finalizan únicamente con instrucciones de Transact-SQL o también con funciones y métodos de la API?
11. ¿Qué es el registro de transacciones y para qué se utiliza?
12. ¿Es cierto que los cursores permiten situarse en una fila específica de un conjunto de resultados devuelto ante una consulta?
13. ¿Puede una aplicación pedir cursores mezclando llamadas utilizando tanto la API como Transact-SQL?
14. Enumere los cuatro tipos de cursores que admite SQL Server
15. Explique, de forma resumida para qué se utilizan los bloqueos.

### 2.2. DISEÑO DE BASES DE DATOS.

1. ¿Qué siete consideraciones acerca del diseño de una Base de Datos conviene tener en cuenta?. Resuma cada una de ellas con una única palabra. Apúntelo en la hoja de referencia (Ver *Crear y mantener bases de datos*).
2. ¿Qué es un Plan de Base de Datos?. ¿Sirve para algo?. Indique de forma razonada cuál le parece la fase más crítica.

3. Se puede dividir el tipo de aplicaciones de BD en dos categorías. Definalas brevemente y ponga un ejemplo concreto para cada una de ellas.
4. Sobre la normalización, sabemos que una clave primaria no acepta valores NULL, pero, ¿tiene importancia que en otros campos de la tabla aparezcan valores NULL?
5. ¿Qué dos cambios, sobre la agregación de columnas y la normalización excesiva, sugieren los libros en pantalla del SQL Server que se pueden hacer para mejorar el rendimiento?
6. ¿En el contexto de las consideraciones acerca del diseño de una bases de datos, con qué intención se realizan las operaciones de Mantenimiento?. Enumérelas e indique las ventajas que aportan.

### 2.3. CREACIÓN DE BASES DE DATOS

1. ¿Qué son las particiones sin formato?. ¿Qué relación tienen con las bases de datos?. ¿Tienen ventajas o desventajas respecto a las particiones con formato?.
2. ¿Qué tres tipos de archivos se utilizan para crear una base de datos?. ¿Qué contiene cada uno?
3. Enumere tres opciones diferentes mediante las que se puede crear una base de datos con SQL Server 2000. Explique brevemente qué es cada una de estas herramientas y cuándo conviene o se pueden utilizar.
4. Cree una nueva base de datos llamada PRUEBA. Observe y comprenda las distintas opciones que se ofrecen al crearla.
5. Desde el Administrador Corporativo, abra el cuadro de diálogo de **Propiedades** de la base de datos que ha creado. Observe las pestañas de *Opciones* y *Permisos*. En la pestaña de *Opciones* averigüe para qué se utiliza cada uno de los valores seleccionados.
6. Mire a ver cómo se crea un usuario nuevo, y hágalo si puede.
7. Visualice el cuadro de diálogo **Propiedades** para un usuario y observe el cuadro de permisos. Explique cómo se indica si se ha concedido un permiso o si se ha denegado.
8. Indique cuál es la instrucción de Transact-SQL que permite crear una base de datos.
9. Elimine la base de datos *prueba*.
10. Cree la base de datos *calificaciones*.

A partir del enunciado del Problema 1, dibuja el esquema ER y luego obtenga el esquema relacional. Se reproduce el enunciado a continuación:

Se desea diseñar una base de datos para un centro de enseñanza que contenga información sobre los alumnos, las asignaturas y las





# PRÁCTICA 4.

## BASE DE DATOS “LIGAINTERNA”

### 1. OBJETIVOS.

Al finalizar esta práctica el alumno será capaz de:

1. Crear una base de datos utilizando el Administrador Corporativo.
2. Obtener una base de datos llamada “LigaInterna” a partir de las notas tomadas.
3. Explicar las ventajas que se obtienen al utilizar del diagrama E/R en la realización de la base de datos.
4. Diferenciar entre el modelado de los datos y el análisis funcional de los datos.

### 2. DETALLES.

#### 2.1. SITUACIÓN Y ACLARACIONES.

##### 2.1.1. Situación del problema.

Una Universidad organiza una competición interna entre equipos de distintas modalidades y necesita realizar una base de datos que le permita gestionar toda la información relacionada con los jugadores, los equipos y los partidos.

Para obtener los detalles del problema nos hemos entrevistado con una de las personas ocupadas de gestionar la información. Se ha tomado nota de la conversación mantenida con dicha persona y una transcripción “literal” aparece en el apartado 2.2.: “Enunciado del problema”.

En una próxima reunión será necesario presentar un prototipo de esta base de datos, por lo que, en este caso, cualquier dato que falte o información que esté incompleta habrá que *imaginarla, suponerla o inventarla* a partir de la información que el analista-diseñador tenga del dominio del problema.

### 2.1.2. Aclaraciones.

La persona que ha realizado la descripción del problema no conoce los conceptos fundamentales del diseño de bases de datos, pero, se ha tenido la suerte de que es un técnico con conocimientos en informática. Esto “significa” que si bien el enunciado es *funcionalmente tendencioso* y las cosas pueden no ser lo que parecen, la descripción de esta persona es mucho más fácilmente abordable, de cara a un modelado de los datos, que si dicha descripción estuviera realizada por personal no técnico.

## 2.2. ENUNCIADO DEL PROBLEMA.

El problema es el siguiente:

Existen varias modalidades deportivas: Fútbol, Fútbol Sala, Balonmano, etc. Para cada una de ellas hay categoría masculina y femenina. He considerado modalidades diferentes a cada una de las modalidades deportivas en categoría masculina y femenina, así por Ej. "Fútbol Sala Masculino" es diferente modalidad que "Fútbol Sala Femenino".

Dentro de cada categoría tenemos varias divisiones, generalmente 1ª, 2ª y 3ª. Dentro de cada división existen grupos. El número de grupos por división es variable dependiendo de que exista un mayor o menor número de equipos en cada grupo.

Cada grupo consta de varios equipos. El número máximo de equipos por grupo es de 14, pudiendo éste ser menor.

Es necesario almacenar información sobre cada equipo con datos como: nombre, número de partidos jugados, empatados, perdidos, no presentado, tantos a favor, etc.

Cada equipo consta de varios jugadores, de los cuales se almacena su información personal. Un jugador puede jugar en varios equipos, siempre y cuando estos equipos pertenezcan a diferentes modalidades.

Cada equipo puede recibir varias sanciones. Una sanción a equipo deberá ir acompañada de la información relativa al tipo de sanción y la jornada en que se impuso dicha sanción.

Cada jugador también podrá recibir varias sanciones. Una sanción a un jugador deberá ir acompañada de información análoga a la que acompaña al tipo de sanción anterior. Estas sanciones deberán ir ligadas también al equipo con el cual el jugador recibió la sanción.

La competición se realiza disputándose varios partidos entre equipos dentro de cada grupo en cada jornada. Es indispensable poder generar la planificación de partidos para cada jornada, es decir definir dentro de cada grupo qué equipos jugarán contra qué equipos. Es interesante también almacenar los resultados de todos y cada uno de los partidos disputados. También es necesario realizar informes para imprimir o incluso publicar en el web las posiciones de clasificación de cada uno de

los equipos dentro de cada grupo, acompañado de la información de número de puntos y número de partidos ganados, empatados, perdidos, etc.”

### **3. TAREAS A REALIZAR.**

Se pide:

1. Proponer un esquema relacional para el problema anterior. Indicar qué tablas, relaciones y restricciones son necesarias a partir de la lectura del problema anterior. Dibujar un esquema relacional SIN CREAR ANTES UN DIAGRAMA ENTIDAD RELACIÓN.
2. Realizar un diagrama E/R del problema anterior.
  - a. Indicar la semántica no recogida.
3. Reducir el diagrama E/R a un esquema relacional.
  - a. Explicar cómo se controla la semántica no recogida en el diagrama E/R.
4. Implementar la solución creando una base de datos con el Administrador corporativo.
  - a. Crear un diagrama que recoja las tablas y sus relaciones.
  - b. Añadir en cada tabla, además de los campos de clave primaria, al menos tres campos más (atributos) prestando especial atención al tipo de dato elegido. Poner para cada campo al menos un ejemplo de los valores que puede contener.’
  - c. Indicar cómo se podrían realizar los informes para publicar en el Web las clasificaciones de cada uno de los equipos.



# PRÁCTICA 5.

## BASE DE DATOS “EXOTICFOOD” (I)

### 1. OBJETIVOS.

Al finalizar esta práctica el alumno será capaz de:

1. Crear una base de datos utilizando el Administrador Corporativo.
2. Crear tablas y asignar el tipo de datos apropiado a cada campo.
3. Crear un diagrama de una base datos.
4. Definir claves primarias.
5. Definir restricciones de unicidad.
6. Especificar atributos NOT NULL.
7. Establecer las restricciones de integridad referencial.

### 2. DETALLES.

#### 2.1. DESCRIPCIÓN GENERAL.

Se desea disponer de una base de datos para una organización que importa y exporta comidas exóticas por todo el mundo.

La empresa almacena diversos datos (se enumeran posteriormente) de los empleados y de los clientes, así como el nombre y el teléfono de las empresas que realizan los transportes. Cuando un cliente realiza un pedido se guarda constancia, además de los datos específicos de dicho pedido (enumerados más adelante), del cliente y el empleado que han participado y también de la empresa de transporte quedando reflejada ésta en el campo de “método de envío”.

En cuanto a los productos, además de los datos específicos, se guarda para cada uno de ellos el proveedor que lo suministra y la categoría a la que pertenece.

Para cada pedido suele obtenerse un informe de detalles de pedidos, en el que se incluye, además del producto y el pedido, el precio unitario, la cantidad y el descuento.

Los empleados de la empresa se dividen en dos categorías laborales: jefes y subordinados. Como para la empresa el coste de almacenamiento es un factor importante, se ha decidido reducir el número de tablas de la base de datos. Debido a esto, se decide que los empleados se almacenarán en una única tabla.

## 2.2. DESCRIPCIÓN DE LOS DATOS.

Hay información que aparece sin ejemplos ya sea porque se encuentran en la descripción de otra entidad o porque no parece necesario al considerarse su contenido evidente (ej.. nombre, apellido, ....).

Habitualmente, cuando se encuentren varias líneas en la casilla de ejemplos, cada una de ellas corresponde con un posible caso diferente.

Se desea disponer de la siguiente información:

### 2.2.1. Relacionados con los productos.

#### PRODUCTOS

| Información                     | Ejemplos  |
|---------------------------------|---|
| Nombre del producto             | Apasionada salsa de chile de Louisiana<br>Mozzarella di Giovanni<br>Original Frankfurter grüne Soße |
| Proveedor que lo suministra     |   |
| Categoría del producto          |   |
| Cantidad por unidad de producto | 16 paquetes de Kg.<br>10 cajas x 12 pizzas<br>30 cajas de regalo                                    |
| Precio unitario                 | 43.9000<br>45.6000<br>123.7900  |
| Unidades en stock               | 104<br>61<br>20   |

| Información                       | Ejemplos     |
|-----------------------------------|--------------|
| Unidades pedidas                  | 0<br>0<br>40 |
| Cantidad pedida en nuevos pedidos | 0<br>5<br>25 |
| Suspender el pedido               | Si o no      |

**PROVEEDORES**

| Información             | Ejemplos   |
|-------------------------|--|
| Nombre de la empresa    | Cooperativa de Quesos 'Las Cabras'<br>Nord-Ost-Fisch Handelsgesellschaft mbH<br>New England Seafood Cannery<br>Plutzer Lebensmittelgroßmärkte AG |
| Nombre Persona contacto | Antonio del Valle Saavedra<br>Petra Winkler  |
| Cargo persona contacto  | International Marketing Mgr.<br>Coordinator Foreign Markets<br>Sales Representative  |
| Dirección               | Order Processing Dept. 2100 Paul Revere Blvd.<br>471 Serangoon Loop, Suite #402<br>Lyngbysild Fiskebakken 10                                     |
| Ciudad                  | Boston<br>Singapore<br>Lyngby  |
| Región                  | Asturias<br><i>desconocida</i>   |
| Código Postal           | S-345 67<br>5442   |
| País                    | Denmark<br>Netherlands   |

| Información | Ejemplos  |
|-------------|---|
| Teléfono    | 08-123 45 67<br>(1) 03.83.00.68<br>(617) 555-3267   |
| Fax         | <i>desconocido</i><br>(1) 03.83.00.62   |
| HomePage    | <i>desconocida</i><br>G'day Mate (on the World Wide Web)# <a href="http://www.microsoft.com/accessdev/sampleapps/gdaymate.htm#">http://www.microsoft.com/accessdev/sampleapps/gdaymate.htm#</a> |

### CATEGORÍAS DE PRODUCTOS

| Información            | Ejemplos   |
|------------------------|--|
| Nombre de la categoría | Bebidas<br>Condimentos<br>Mariscos                                 |
| Descripción            | Salsas dulces y sabrosas, condimentos, comidas para untar y aliños |
| Imagen                 | Una imagen representativa de la categoría del producto             |

#### 2.2.2. Relacionados con los empleados.

### EMPLEADOS

| Información             | Ejemplos  |
|-------------------------|---|
| Primer apellido         |   |
| Nombre                  |   |
| Cargo                   | Vicepresidente, ventas<br>Representante de ventas           |
| Tratamiento de cortesía | Dr., Ms., Mrs., Mr., Don, Doña, ...                         |
| Fecha nacimiento        |   |
| Fecha del contrato      |   |
| Dirección               | Coventry House, Miner Rd.<br>Edgeham Hollow, Winchester Way |



| Información            | Ejemplos   |
|------------------------|--|
| Ciudad                 |  |
| Región                 |  |
| Código postal          | RG1 9SP<br>98105<br>WG2 7LT  |
| País                   |  |
| Teléfono de casa       | (206) 555-9857   |
| Extensión              | 5467   |
| Foto                   |  |
| Observaciones          | Su formación incluye una licenciatura en sicología por la Universidad de Navarra en 1985. También tiene diversos cursos y es miembro de diferentes sociedades científicas.       |
| Presenta informes a... | Persona que es su jefe y a la que le presenta los informes   |
| Ruta (path) de la foto | <a href="http://accweb/emmployees/davolio.bmp">http://accweb/emmployees/davolio.bmp</a><br><a href="http://accweb/emmployees/fuller.bmp">http://accweb/emmployees/fuller.bmp</a> |

Se desea guardar para cada empleado una lista de los territorios que están a su cargo. La información sobre territorios será, además del nombre del territorio (hay unos 50 distintos), la región a la que dicho territorio pertenece. Esta región puede verse como un atributo multivalorado de la entidad territorios, ya que, de momento, se pueden encontrar únicamente cuatro regiones distintas.

## TERRITORIOS

| Información               | Ejemplos                            |
|---------------------------|-------------------------------------|
| Nombre del territorio     | Philadelphia<br>Neward<br>Rockville |
| Región a la que pertenece |                                     |

### 2.2.3. Relacionados con los pedidos.

#### PEDIDOS

| Información                   | Ejemplos   |
|-------------------------------|--|
| Identificación del pedido     |  |
| El cliente que lo pide        |  |
| Empleado que lo tramita       |  |
| Fecha de pedido               |  |
| Fecha de entrega comprometida |  |
| Fecha de envío                |  |
| Empresa que realiza el envío  | Será una de las empresas de transportes con la que se trabaja habitualmente            |
| Costes del envío              | El dinero que cuesta realizar el envío. Es una cantidad que no suele superar los 900\$ |
| Nombre a quien se envía       | Ana Trujillo Emparedados y helados<br>Berglunds snabbköp<br>Save-a-lot Markets         |
| Dirección de envío            | Avda. de la Constitución 2222<br>Berguvsvägen 8<br>187 Suffolk Ln.                     |
| Ciudad de envío               | Rio de Janeiro<br>München<br>Butte   |
| Región de envío               | Québec<br><i>desconocida</i><br>Isle of Wight  |
| Código postal de envío        | PO31 7PJ<br>02389-890<br>80805   |
| País de envío                 | Germany<br>USA<br>Mexico   |

**CLIENTES**

| Información                      | Ejemplos   |
|----------------------------------|--|
| Nombre de la empresa             | FISSA Fabrica Inter. Salchichas S.A.<br>Folies gourmandes<br>Folk och få HB  |
| Nombre de la persona de contacto | Karl Jablonski<br>Matti Karttunen<br>Zbyszek Piestrzeniewicz                 |
| Cargo persona contacto           | Propietario<br>Director de ventas<br>Propietario / Asistente de marketing    |
| Dirección                        | Carrera 22 con Ave. Carlos Soublette #8-35<br>City Center Plaza 516 Main St. |
| Ciudad                           | Rio de Janeiro<br>San Cristóbal  |
| Región                           | RJ<br>Táchira<br><i>desconocida</i>  |
| Código postal                    | <i>desconocido</i><br>PO31 7PJ<br>14776                                      |
| País                             | France<br>Canada<br>USA  |
| Teléfono                         | (198) 555-8888<br>0555-09876<br>30.59.84.10                                  |
| Fax                              | (503) 555-2376<br>2967 3333<br><i>desconocido</i>                            |

**EMPRESAS DE TRANSPORTE**

| Información          | Ejemplos                           |
|----------------------|------------------------------------|
| Nombre de la empresa | United Package<br>Federal Shipping |
| Teléfono             | (503) 555-9831<br>(503) 555-3199   |

**DETALLES DE PEDIDOS**

| Información     | Ejemplos  |
|-----------------|---|
| El pedido       |   |
| El producto     |   |
| Precio unitario | Un valor normalmente no superior a 200\$        |
| Cantidad        | Un valor normalmente no superior a 100 unidades |
| Descuento       | 0<br>0,2<br>0,25                                |

**3. SE PIDE.**

1. Realizar un diagrama Entidad – Relación de la base de datos que podría utilizarse para almacenar la información anterior.
2. Obtener el esquema relacional.
3. Observar si es posible eliminar alguna de las tablas obtenidas.
4. Crear las tablas necesarias, utilizando el administrador corporativo, atendiendo de forma especial a:
  - a. El tipo de datos más indicado para cada uno de sus campos (según sean los datos que se van a almacenar en cada campo y de la longitud necesaria para hacerlo).
  - b. Observando si pueden o no admitir valores NULL
  - c. Indicando si son columnas identidad y estableciendo un valor de inicialización y otro de incremento.
  - d. Definiendo las claves primarias cuando sea necesario.
  - e. Estableciendo restricciones de unicidad en los campos que lo necesiten.
  - f. Definiendo las claves externas cuando sea necesario.
5. Obtener un diagrama de la base de datos creada.

Nota: si crees que falta algún dato imagínate que te entrevistas con el cliente y, a partir de la información que él te facilite, propón una solución.



# PRÁCTICA 6.

## BASE DE DATOS “EXOTICFOOD” (II)

### **1. OBJETIVOS.**

Al finalizar esta práctica el alumno será capaz de:

1. Crear una base de datos utilizando el Administrador Corporativo.
2. Seleccionar el tipo de dato más apropiado para cada campo.
3. Dimensionar correctamente el tamaño de los campos.
4. Elegir la clave primaria más apropiada para cada tabla.
5. Definir restricciones de unicidad.
6. Especificar atributos NOT NULL.
7. Establecer las restricciones de integridad referencial.

### **2. DETALLES.**

#### **2.1. OPCIONES EN EL DISEÑO DE LA BASE DE DATOS.**

En el diseño realizado en la práctica anterior tuvieron que surgirte dudas en diversos momentos. Independientemente de la solución adoptada, habrás tenido que considerar las siguientes cuestiones que en esta práctica explicarás detalladamente.

**2.1.1. Entidades: CATEGORÍAS, PRODUCTOS y relación CAT-PRO.**

1. ¿Puede un producto pertenecer a más de una categoría?  
Dibuja el diagrama E/R correspondiente para cada caso. Dibuja las tablas que se obtendrían al reducir cada uno de los diagramas anteriores. Explica las restricciones. Observa si existe alguna posible reducción.
2. ¿Va a existir algún producto que no tenga categoría?  
Dibuja el diagrama E/R correspondiente para cada caso, combinando las dos posibilidades con las de la pregunta anterior (4 casos). Dibuja las tablas que se obtendrían al reducir cada uno de los diagramas anteriores. Explica las restricciones. Observa si existe alguna posible reducción.

**2.1.2. PROVEEDORES, PRODUCTOS; PROVEE-PRODUC.**

1. ¿Un producto puede ser suministrado por más de un proveedor?  
Dibuja el diagrama E/R correspondiente para cada caso. Dibuja las tablas que se obtendrían al reducir cada uno de los diagramas anteriores. Explica las restricciones. Observa si existe alguna posible reducción.
2. ¿Todos los productos tendrán proveedor?  
Dibuja el diagrama E/R correspondiente para cada caso, combinando las dos posibilidades con las de la pregunta anterior (4 casos). Dibuja las tablas que se obtendrían al reducir cada uno de los diagramas anteriores. Explica las restricciones. Observa si existe alguna posible reducción.

**2.1.3. PRODUCTOS; PEDIDOS, DETALLES PEDIDOS.**

1. ¿Un pedido puede tener más de un producto?  
Contesta y explica la respuesta.
2. ¿Puede haber pedidos sin productos?  
Contesta y explica la respuesta.
3. ¿Puede ser NULL PedID en DETALLES PEDIDOS?  
Contesta y explica la respuesta.
4. ¿Interesa poner un DetPedID?  
Contesta y explica la respuesta.

**2.1.4. EMPLEADOS.**

1. ¿Cuál es la clave primaria?  
Contesta y explica la respuesta.
2. Relación “Informa A”.  
Explica qué significa y cómo se resuelve.
3. ¿Puede tener más de un jefe un subordinado?  
Contesta e indica cómo se resolvería en caso afirmativo.



**2.1.5. CLIENTES, EMPLEADOS, TRANSPORTISTA; PEDIDOS.**

1. ¿Cuáles son las claves primarias?  
Contesta y explica la respuesta.
2. Indica si los siguientes campos aceptan NULL y explica por qué.  
CliID, EmpID, MétodoEnvío (Empresa que realiza el envío en PEDIDOS).

**2.1.6. EMPLEADOS, TERRITORIOS, REGIONES; TER-EMP.**

1. ¿Clave primaria de TERRITORIOS?  
Contesta y explica la respuesta.
2. ¿Un empleado puede llevar varios territorios?  
Dibuja el diagrama E/R correspondiente para cada caso. Dibuja las tablas que se obtendrían al reducir cada uno de los diagramas anteriores. Explica las restricciones. Observa si existe alguna posible reducción.
3. ¿Un empleado lleva, al menos, un territorio?  
Dibuja el diagrama E/R correspondiente para cada caso, combinando las dos posibilidades con las de la pregunta anterior (4 casos). Dibuja las tablas que se obtendrían al reducir cada uno de los diagramas anteriores. Explica las restricciones. Observa si existe alguna posible reducción.
4. ¿A cuantas regiones puede pertenecer un territorio?  
Contesta y explica la respuesta.

**2.2. TIPOS DE DATOS UTILIZADOS.**

Suponiendo que en el diseño que has realizado te han aparecido las siguientes tablas y campos, indica qué tipo de dato has seleccionado en cada caso, otras alternativas posibles, su longitud y si admite o no valores NULL.

**2.2.1. Relacionados con los productos.****PRODUCTOS**

| Nombre Campo | Tipo de datos y explicación |
|--------------|-----------------------------|
| ProID        |                             |
| ProNombre    |                             |
| ProveeID     |                             |
| CatID        |                             |
| CantiPorUni  |                             |
| PrecioUni    |                             |
| UniStock     |                             |

| Nombre Campo | Tipo de datos y explicación |
|--------------|-----------------------------|
| UniPedida    |                             |
| NuevosPed    |                             |
| SuspenPed    |                             |

**PROVEEDORES**

| Nombre Campo | Tipo de datos y explicación |
|--------------|-----------------------------|
| ProveeID     |                             |
| ContacNom    |                             |
| ContacCar    |                             |
| Direcc       |                             |
| Ciudad       |                             |
| Region       |                             |
| CodPostal    |                             |
| Pais         |                             |
| Telfno       |                             |
| Fax          |                             |
| HomPag       |                             |

**CATEGORÍAS DE PRODUCTOS**

| Nombre Campo | Tipo de datos y explicación |
|--------------|-----------------------------|
| CatID        |                             |
| CatNombre    |                             |
| CatDescrip   |                             |
| CatImagen    |                             |

**2.2.2. Relacionados con los empleados.****EMPLEADOS**

| Nombre Campo | Tipo de datos y explicación |
|--------------|-----------------------------|
| EmpID        |                             |
| ENombre      |                             |
| ECargo       |                             |

| Nombre Campo | Tipo de datos y explicación |
|--------------|-----------------------------|
| ETratam      |                             |
| EFecNac      |                             |
| EFecContra   |                             |
| EDirecc      |                             |
| ECiudad      |                             |
| ERegion      |                             |
| ECodPos      |                             |
| EPais        |                             |
| ETfnCas      |                             |
| EExtens      |                             |
| EFoto        |                             |
| EObserv      |                             |
| EInformaA    |                             |
| ERutFot      |                             |

**EMP-TERR**

| Nombre Campo | Tipo de datos y explicación |
|--------------|-----------------------------|
| EmpID        |                             |
| TerID        |                             |

**TERRITORIOS**

| Nombre Campo | Tipo de datos y explicación |
|--------------|-----------------------------|
| TerID        |                             |
| TerNombre    |                             |
| TerRegID     |                             |

**REGIONES**

| Nombre Campo | Tipo de datos y explicación |
|--------------|-----------------------------|
| RegID        |                             |
| RNombre      |                             |

### 2.2.3. Relacionados con los pedidos.

#### PEDIDOS

| Nombre Campo | Tipo de datos y explicación |
|--------------|-----------------------------|
| PedID        |                             |
| PCliID       |                             |
| PEmpID       |                             |
| PFecPed      |                             |
| PFecCompro   |                             |
| PFecEnv      |                             |
| PTraID       |                             |
| PCosEnv      |                             |
| PNomEnv      |                             |
| PDireEnv     |                             |
| PCiuEnv      |                             |
| PRegEnv      |                             |
| PCodPosEnv   |                             |
| PPaiEnv      |                             |

#### CLIENTES

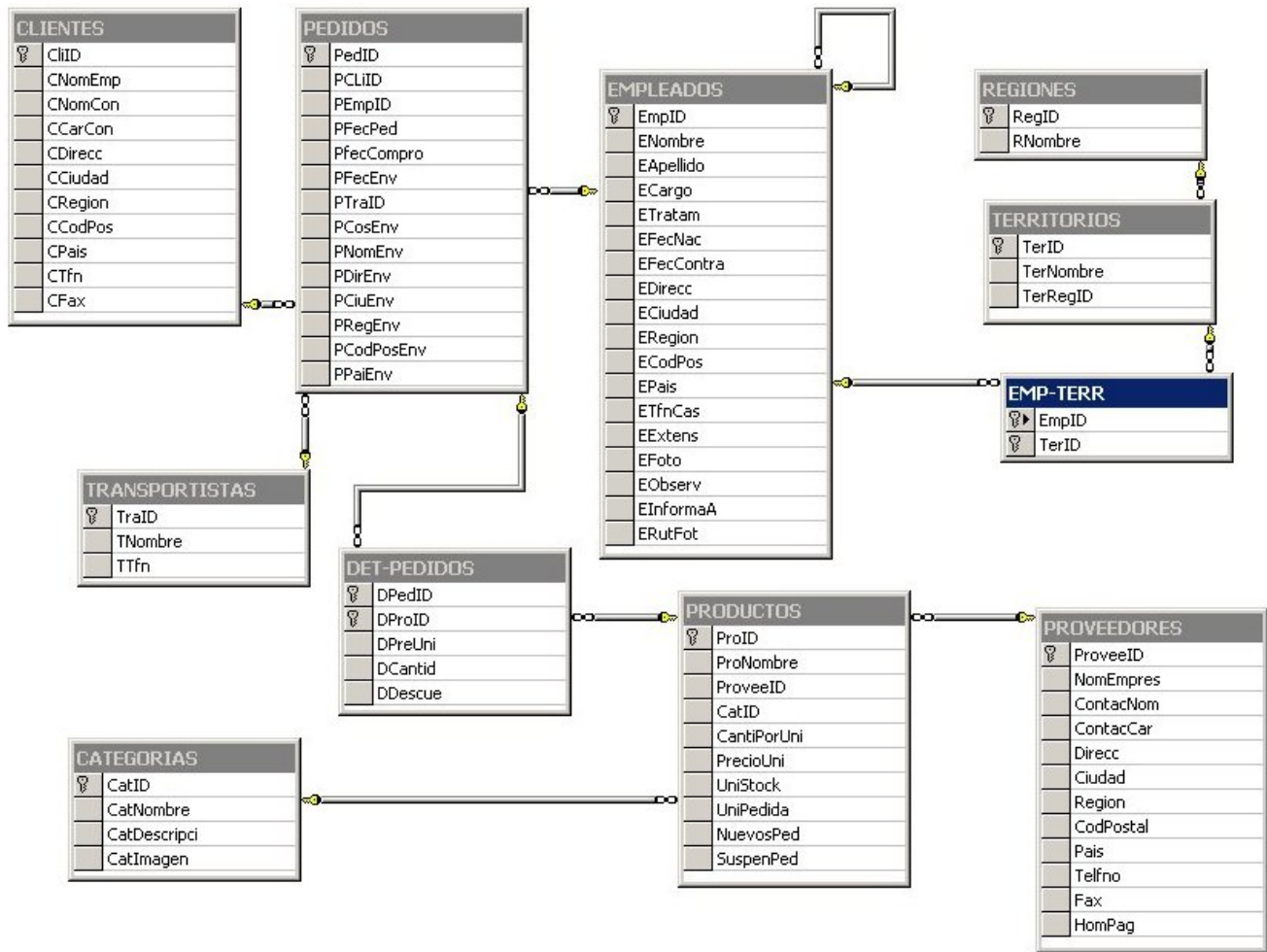
| Nombre Campo | Tipo de datos y explicación |
|--------------|-----------------------------|
| CliID        |                             |
| CNomCon      |                             |
| CCarCon      |                             |
| CDirecc      |                             |
| CCiudad      |                             |
| CRegion      |                             |
| CCodPos      |                             |
| CPais        |                             |
| CTfn         |                             |
| CFax         |                             |

**TRANSPORTISTAS**

| Nombre Campo | Tipo de datos y explicación |
|--------------|-----------------------------|
| TraID        |                             |
| TNombre      |                             |
| TTfn         |                             |

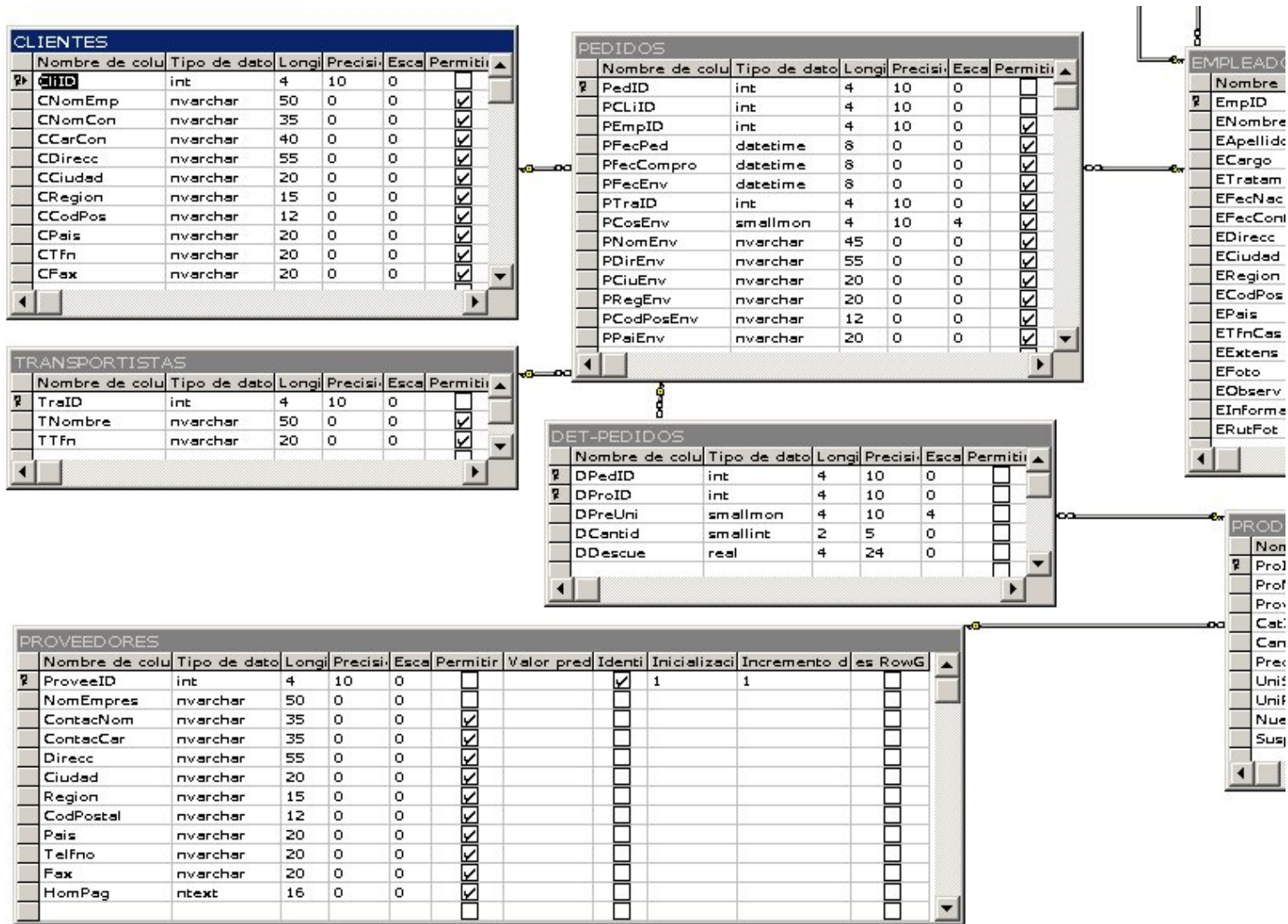
**DET-PEDIDOS**

| Nombre Campo | Tipo de datos y explicación |
|--------------|-----------------------------|
| DPedID       |                             |
| DProID       |                             |
| DPreUni      |                             |
| DCantid      |                             |
| DDescue      |                             |



### 3. RELACIONES “EXOTICFOOD”.

#### 4. TIPOS DE DATOS.



| EMPLEADOS      |              |       |         |      |         |  |
|----------------|--------------|-------|---------|------|---------|--|
| Nombre de colu | Tipo de dato | Longi | Precisi | Esca | Permiti |  |
| EmpID          | int          | 4     | 10      | 0    |         |  |
| ENombre        | nvarchar     | 15    | 0       | 0    |         |  |
| EApellido      | nvarchar     | 20    | 0       | 0    |         |  |
| ECargo         | nvarchar     | 30    | 0       | 0    |         |  |
| ETratam        | nvarchar     | 4     | 0       | 0    |         |  |
| EFecNac        | datetime     | 8     | 0       | 0    |         |  |
| EFecContra     | datetime     | 8     | 0       | 0    |         |  |
| EDirecc        | nvarchar     | 55    | 0       | 0    |         |  |
| ECiudad        | nvarchar     | 20    | 0       | 0    |         |  |
| ERegion        | nvarchar     | 15    | 0       | 0    |         |  |
| ECodPos        | nvarchar     | 12    | 0       | 0    |         |  |
| EPais          | nvarchar     | 20    | 0       | 0    |         |  |
| ETfnCas        | nvarchar     | 20    | 0       | 0    |         |  |
| EExtens        | nvarchar     | 4     | 0       | 0    |         |  |
| EFoto          | image        | 16    | 0       | 0    |         |  |
| EObserv        | ntext        | 16    | 0       | 0    |         |  |
| EInformaA      | int          | 4     | 10      | 0    |         |  |
| ERutFot        | nvarchar     | 150   | 0       | 0    |         |  |

| REGIONES       |              |       |         |      |         |  |
|----------------|--------------|-------|---------|------|---------|--|
| Nombre de colu | Tipo de dato | Longi | Precisi | Esca | Permiti |  |
| RegID          | int          | 4     | 10      | 0    |         |  |
| RNombre        | nvarchar     | 20    | 0       | 0    |         |  |

| TERRITORIOS    |              |       |         |      |         |  |
|----------------|--------------|-------|---------|------|---------|--|
| Nombre de colu | Tipo de dato | Longi | Precisi | Esca | Permiti |  |
| TerID          | int          | 4     | 10      | 0    |         |  |
| TerNombre      | nvarchar     | 25    | 0       | 0    |         |  |
| TerRegID       | int          | 4     | 10      | 0    |         |  |

| EMP-TERR       |              |       |         |      |         |  |
|----------------|--------------|-------|---------|------|---------|--|
| Nombre de colu | Tipo de dato | Longi | Precisi | Esca | Permiti |  |
| EmpID          | int          | 4     | 10      | 0    |         |  |
| TerID          | int          | 4     | 10      | 0    |         |  |

| PRODUCTOS      |              |       |         |      |         |  |
|----------------|--------------|-------|---------|------|---------|--|
| Nombre de colu | Tipo de dato | Longi | Precisi | Esca | Permiti |  |
| ProID          | int          | 4     | 10      | 0    |         |  |
| ProNombre      | nvarchar     | 50    | 0       | 0    |         |  |
| ProveeID       | int          | 4     | 10      | 0    |         |  |
| CatID          | int          | 4     | 10      | 0    |         |  |
| CantiPorUni    | nvarchar     | 25    | 0       | 0    |         |  |
| PrecioUni      | smallmon     | 4     | 10      | 4    |         |  |
| UniStock       | smallint     | 2     | 5       | 0    |         |  |
| UniPedida      | smallint     | 2     | 5       | 0    |         |  |
| NuevosPed      | smallint     | 2     | 5       | 0    |         |  |
| SuspenPed      | bit          | 1     | 0       | 0    |         |  |

| CATEGORIAS     |              |       |         |      |         |  |
|----------------|--------------|-------|---------|------|---------|--|
| Nombre de colu | Tipo de dato | Longi | Precisi | Esca | Permiti |  |
| CatID          | int          | 4     | 10      | 0    |         |  |
| CatNombre      | nvarchar     | 15    | 0       | 0    |         |  |
| CatDescripci   | ntext        | 16    | 0       | 0    |         |  |
| CatImagen      | image        | 16    | 0       | 0    |         |  |



# PRÁCTICA 7. “EXOTICFOOD” (III).

## RESTRICCIONES, VISTAS E ÍNDICES

### 1. OBJETIVOS.

Al finalizar esta práctica el alumno será capaz de:

1. Explicar la utilidad de las restricciones CHECK
2. Crear restricciones CHECK utilizando el Administrador Corporativo.
3. Explicar la utilidad de las vistas.
4. Crear vistas utilizando el Administrador Corporativo.
5. Explicar la utilidad de los índices.
6. Enumerar y explicar los diferentes tipos de índices que puede tener una tabla.
7. Crear índices utilizando el Administrador Corporativo.

### 2. DETALLES.

#### 2.1. RESTRICCIONES DE DOMINIO.

##### 2.1.1. Restricciones CHECK (*información procedente de los Libros en Pantalla*).

Las restricciones de comprobación (CHECK) exigen la integridad del dominio mediante la limitación de los valores que puede aceptar una columna. Son similares a las restricciones FOREIGN KEY porque controlan los valores que se colocan en una columna. La diferencia estriba en la forma en que determinan los valores válidos: Las restricciones FOREIGN KEY obtienen la lista de valores válidos de

otra tabla, mientras que las restricciones CHECK determinan los valores válidos a partir de una expresión lógica que no se basa en datos de otra columna. Por ejemplo, es posible limitar el intervalo de valores para una columna **salary** (salario) mediante la creación de una restricción CHECK que permita únicamente datos del intervalo comprendido entre -si por ejemplo se tratara de dólares- \$15.000 y \$100.000. Así se impide que se escriban salarios fuera de este intervalo.

Puede crear una restricción CHECK con cualquier expresión lógica (booleana) que devuelva TRUE (verdadero) o FALSE (falso) basándose en operadores lógicos. Para el ejemplo anterior, la expresión lógica sería:

```
salary >= 15000 AND salary <= 100000
```

Es posible aplicar varias restricciones CHECK a una sola columna. Estas restricciones se evalúan en el orden en el que fueron creadas. También es posible aplicar una sola restricción CHECK a varias columnas si se crea al final de la tabla. Por ejemplo, una restricción CHECK para varias columnas se puede utilizar para confirmar que cualquier fila con un valor EE.UU. en la columna **country** (país) tiene también un valor de dos caracteres en la columna **state** (estado). Así se pueden comprobar varias condiciones en un mismo sitio.

Ejemplos de restricciones CHECK de la base de datos pubs:

1) Nivel máximo de un puesto de trabajo (Pubs, jobs)

```
([max_lvl] <= 250)
```

2) Restricción en los valores y el formato que puede introducirse como identificadores de autor: (xxx-xx-xxxx), siendo x un número de 0 a 9.

```
(([au_id] like '[0-9][0-9][0-9]-[0-9][0-9]-[0-9][0-9][0-9][0-9]'))
```

3) Restricción en los valores y el formato que puede introducirse como identificadores de empleado.

```
(([emp_id] like '[A-Z][A-Z][A-Z][1-9][0-9][0-9][0-9][0-9][FM]') or ([emp_id] like '[A-Z]-[A-Z][1-9][0-9][0-9][0-9][0-9][FM]'))
```

4) Restricción en los valores y el formato que puede introducirse como identificadores de publicaciones.

```
([pub_id] = '1756' or [pub_id] = '1622' or [pub_id] = '0877' or [pub_id] = '0736' or [pub_id] = '1389' or ([pub_id] like '99[0-9][0-9]'))
```

### 2.1.2. Creación de restricciones CHECK.

Utilizando el Administrador Corporativo, crea las siguientes restricciones en las tablas que se indican, poniendo un nombre descriptivo a la restricción.

## PRODUCTOS

- El precio unitario deber ser mayor o igual que 0.
- La cantidad pedida en los nuevos pedidos ha de ser mayor o igual que cero
- Las unidades en stock han de ser mayor o igual a 0.
- Las unidades pedidas han de ser mayor o igual a cero.

## EMPLEADOS

- La fecha de nacimiento ha de ser anterior a la fecha actual.

## DET-PEDIDOS

- El descuento ha de ser mayor o igual que cero y menor o igual que 1.
- La cantidad pedida ha de ser mayor que cero.
- El precio unitario ha de ser mayor o igual que cero.

## 2.2. CREACIÓN DE VISTAS.

### 2.2.1. Información procedente de los Libros en Pantalla.

Las vistas suelen utilizarse para centrar, simplificar y personalizar la percepción de la base de datos para cada usuario. Pueden emplearse como mecanismos de seguridad al permitir a los usuarios tener acceso a los datos mediante la vista sin concederles permisos para que tengan acceso directo a las tablas base subyacentes de la vista. Las vistas también pueden utilizarse cuando se copian datos en o desde Microsoft® SQL Server™, así como para realizar particiones de datos.

Antes de crear una vista, hay que tener en cuenta las siguientes indicaciones:

- Sólo puede crear vistas en la base de datos actual. Sin embargo, las tablas y las vistas a las que se haga referencia desde la nueva vista pueden encontrarse en otras bases de datos e, incluso, en otros servidores, si la vista se define mediante consultas distribuidas.
- Los nombres de las vistas deben seguir las reglas para los identificadores y ser únicos para cada usuario. Además, el nombre debe ser distinto del de cualquier tabla de las que el usuario sea propietario.
- Puede generar vistas dentro de otras vistas y en procedimientos que hagan referencia a vistas. Microsoft® SQL Server™ permite anidar hasta 32 niveles de vistas.
- No puede asociar reglas, definiciones DEFAULT ni desencadenadores con las vistas.
- La consulta que defina a la vista no puede incluir las cláusulas ORDER BY, COMPUTE o COMPUTE BY, ni la palabra clave INTO.
- No puede generar índices ni crear definiciones de índices de texto en las vistas.
- No puede crear vistas temporales, ni vistas dentro de tablas temporales.
- No puede emitir consultas de texto en una vista, aunque una definición de vista puede incluir una consulta de texto si ésta hace referencia a una tabla configurada para indización de texto.
- Debe especificar el nombre de todas las columnas de la vista en el caso de que:
  - alguna de las columnas de la vista derive de una expresión aritmética, una función integrada o una constante.

- Dos o más columnas de la vista tendrían, en caso contrario, el mismo nombre (normalmente, debido a que la definición de la vista incluye una combinación y las columnas de dos o más tablas diferentes tienen el mismo nombre).
- Desea darle a una columna de la vista un nombre distinto del de la columna de la que deriva. (También puede cambiar el nombre de las columnas en la vista). Una columna de una vista hereda los tipos de datos de la columna de la que deriva, aunque no cambie su nombre.

---

**Nota** Esta regla no se aplica cuando una vista se basa en una consulta que contiene una combinación externa, ya que las columnas pueden cambiar al pasar de no permitir valores nulos a permitirlos.

---

De lo contrario, no necesitará especificar nombres de columnas cuando cree la vista. SQL Server asigna a las columnas de la vista los nombres y tipos de datos de las columnas a las que se hace referencia en la consulta que define a la vista. La lista de selección puede ser una enumeración completa o parcial de los nombres de las columnas de las tablas base.

### 2.2.2. Creación de vistas.

Realiza las siguientes vistas utilizando el Administrador Corporativo:

- 1) Listado de todos los productos, cuyo pedido no se ha suspendido, en el que aparezca además la categoría a la que pertenecen (el nombre de la categoría y no únicamente el ID).
- 2) Listado de los productos que se siguen recibiendo, en el que aparece el identificador del producto, el nombre, el proveedor (nombre del proveedor, no sólo el ID) y el precio unitario.
- 3) Listado en el que aparezca,

*de cada pedido:*

el nombre de la empresa que realiza el transporte y la dirección, ciudad, región, código postal y país de envío;

*de cada cliente:*

el nombre de la empresa, la dirección, ciudad, región, código postal y país, así como el nombre y el apellido en una sola columna;

*de los pedidos:*

el identificador del pedido con su fecha de pedido, fecha comprometida y fecha de envío;

*de la empresa transportista:*

el nombre de la empresa

*de los detalles de pedidos:*

el identificador del producto, el nombre del producto –de la tabla productos-, el precio unitario y la cantidad.

En el encabezado de cada columna aparecerá un nombre descriptivo, y no el nombre del campo.

- 4) Listado en el que aparezcan las siguientes columnas: identificador del pedido, identificador del producto, nombre del producto, precio unitario, cantidad pedida y descuento aplicado, y con otra columna en la que aparezca el precio final de cada pedido, tras aplicarle el descuento.

En el encabezado de cada columna aparecerá un nombre descriptivo, y no el nombre del campo.

## 2.3. ÍNDICES.

### 2.3.1. Información sobre índices.

Leer la información de los Libros en Pantalla relacionada con la creación de índices.

**Contesta a las siguientes preguntas:**

1. Factor de relleno: Para qué sirve, qué valor hay que utilizar, casos en los que interesa utilizar otros valores.
2. ¿Qué relación hay entre los índices y las actualizaciones y las eliminaciones?
3. ¿De qué dos formas se pueden definir índices en SQL?
4. ¿Qué cuatro restricciones acepta una de las instrucciones anteriores, relacionadas con la creación de índices?

### 2.3.2. Crear índices en ExoticFood.

Además de los generados automáticamente al establecer las claves primarias, se crearán los siguientes índices:

#### **PRODUCTOS**

Nombre del producto  
Identificador de la categoría  
Identificador del proveedor

#### **PROVEEDORES**

Nombre empresa proveedora  
Código postal

#### **CATEGORÍAS (DE PRODUCTOS)**

Nombre de la categoría

#### **EMPLEADOS**

Apellido  
Código Postal

**PEDIDOS**

Identificación del cliente  
Identificación del empleado  
Fecha de pedido  
Fecha envío  
Empresa mediante la que se envía  
Código Postal de envío

**CLIENTES**

Ciudad  
Nombre de la empresa  
Código Postal  
Región  
Dirección

**2.3.3. Responde a las siguientes cuestiones:**

- Indica si crees que puede ser interesante añadir o quitar algún índice.
- ¿De los índices anteriores, interesaría que alguno de ellos fuera un único índice compuesto por varias columnas?
- ¿Cómo podrías conocer en qué columnas interesa tener índices y si es conveniente que alguno de los índices esté compuesto por varias columnas?
- ¿Qué herramienta utiliza SQL para optimizar los índices y qué información es necesario proporcionarle a esta herramienta?

## PRÁCTICA 8. “EXOTICFOOD” (IV).

### SQL Y EL ANALIZADOR DE CONSULTAS

#### 1. OBJETIVOS.

Al finalizar esta práctica el alumno, utilizando el Analizador de Consultas será capaz de:

1. Crear una base de datos y tablas.
2. Establecer restricciones de clave primaria y clave ajena.
3. Crear restricciones DEFAULT.
4. Crear restricciones CHECK.
5. Crear vistas.
6. Crear índices.
7. Explicar para qué se emplea ALTER TABLE y será capaz de utilizarla.
8. Realizar consultas.

#### 2. DETALLES.

##### 2.1. CREACIÓN DE TABLAS.

Utilizando el Analizador de Consultas realiza las siguientes operaciones:

(Usuario **sa** y sin contraseña)

1. Crea una nueva base de datos llamada **EF2** (ExoticFood 2 u otro nombre cualquiera).

2. En la base de datos anterior, crea las siguientes tablas (mismo tipo de datos que en ExoticFood):

*Restricciones CHECK.*

Ejemplos de creación de restricciones CHECK en el momento de creación de la tabla (también pueden definirse después, modificando la tabla con ALTER TABLE):

```
CREATE TABLE jobs(
    job_id smallint
        IDENTITY(1,1)
        PRIMARY KEY CLUSTERED,
    job_desc varchar(50) NOT NULL
        DEFAULT 'New Position - title not
        formalized yet',
    min_lvl tinyint NOT NULL
        CHECK (min_lvl >= 10),
    max_lvl tinyint NOT NULL
        CHECK (max_lvl <= 250)
)
```

*También puede ponerse nombre a la restricción:*

```
CONSTRAINT CK_emp_id CHECK (emp_id LIKE
    '[A-Z][A-Z][A-Z][1-9][0-9][0-9][0-9]
    [0-9][FM]' OR
    emp_id LIKE '[A-Z]-[A-Z][1-9][0-9]
    [0-9][0-9][0-9][FM]')
```

*Restricciones DEFAULT*

Ejemplos de creación de restricciones DEFAULT en el momento de creación de la tabla (también pueden definirse después, modificando la tabla con ALTER TABLE):

```
CREATE TABLE employee (
    emp_id empid
        CONSTRAINT PK_emp_id PRIMARY KEY NONCLUSTERED
        CONSTRAINT CK_emp_id CHECK (emp_id LIKE
            '[A-Z][A-Z][A-Z][1-9][0-9][0-9][0-9]
            [0-9][FM]' or
            emp_id LIKE '[A-Z]-[A-Z][1-9][0-9][0-9]
            [0-9][0-9][FM]'),
    /* Each employee ID consists of three characters
    that represent the employee's initials, followed by
    a five digit number ranging from 10000 to 99999 and
    then the employee's gender (M or F). A (hyphen) -
    is acceptable for the middle initial. */
    fname varchar(20) NOT NULL,
    minit char(1) NULL,
    lname varchar(30) NOT NULL,
    job_id smallint NOT NULL DEFAULT 1
```



```

/* Entry job_id for new hires. */
REFERENCES jobs(job_id),
    job_lvl tinyint DEFAULT 10,
/* Entry job_lvl for new hires. */
    pub_id char(4) NOT NULL DEFAULT ('9952')
REFERENCES publishers(pub_id),
/* By default, the Parent Company Publisher is
    the company to whom each employee reports. */
    hire_date datetime NOT NULL DEFAULT (getdate())
/* By default, the current system date will be
    entered. */
)

```

▪ PRODUCTOS.

a. Tendrá las siguientes restricciones CHECK.

- i. El Precio Unitario deber ser mayor o igual que 0.
- ii. La cantidad pedida en los nuevos pedidos ha de ser mayor o igual que cero
- iii. Las unidades en stock han de ser mayor o igual a 0.
- iv. Las unidades pedidas han de ser mayor o igual a cero.

b. Tendrá las siguientes restricciones DEFAULT

Los campos PrecioUni, UniStock, UniPedida, NuevosPed y Suspended tendrán “0” como valor por defecto.

▪ DET-PEDIDOS.

a. Tendrá las siguientes restricciones CHECK.

- i. El descuento ha de ser mayor o igual que cero y menor o igual que 1.
- ii. La cantidad pedida ha de ser mayor que cero.
- iii. El precio unitario ha de ser mayor o igual que cero.

b. Tendrá las siguientes restricciones DEFAULT

- i. El descuento y el precio unitario serán “0” por defecto.
- ii. La cantidad pedida será “1”.

▪ PEDIDOS.

a. Tendrá las siguientes restricciones CHECK.

- i. El código postal de envío se ajustará a la estructura de un código postal de la provincia de León: “24xxx”.

b. Tendrá las siguientes restricciones DEFAULT

- i. El valor por defecto del campo PCosEnv será “0”.
- ii. En el campo PFecPed (fecha de pedido) se pondrá la fecha actual.

▪ CLIENTES.

## 2.2. MODIFICACIÓN DE TABLAS.

1. Crear los siguientes índices:
  - PRODUCTOS.
    - a. Nombre del producto
    - b. Identificador de la categoría
    - c. Identificador del proveedor
  - CLIENTES.
    - a. Ciudad
    - b. Nombre de la empresa
    - c. Código Postal
    - d. Región
2. Realizar los siguientes cambios:
  - En la tabla PRODUCTOS.
    - a. Añadir las columnas:
      - i. *Almacen*: contiene el nombre de la Provincia española en la que se encuentra el almacén para ese producto.
      - ii. *Pasillo*: Número de pasillo, dentro del almacén. La empresa no tiene ningún almacén con más de 300 pasillos.
      - iii. *Estanteria*: El número de la estantería en la que se encuentra el producto. Como máximo hay 200 estanterías por pasillo.
    - b. Modificar y borrar:
      - i. *Estanteria*: Se decide eliminar el campo *pasillo* y numerar las estanterías en consecuencia (todas las estanterías del almacén se identifican mediante un número diferente). Es necesario modificar el tipo de datos de la columna *estanteria*.
      - ii. Borrar la columna *Pasillo*.
      - iii. Añadir, a la tabla PRODUCTOS, una restricción de manera que el número de la estantería siempre sea mayor que 1 y menor o igual que el máximo número posible. Como esta decisión se toma con posterioridad a la introducción de datos, establecerla de manera que no se compruebe con los datos ya existentes.

## 2.3. CREACIÓN DE VISTAS.

1. Crear las siguientes vistas:
  - a. A partir de la tabla PRODUCTOS:
    - i. Nombre del producto, proveedor y precio unitario.
    - ii. Identificador del producto, unidades en stock, unidades pedidas y Nuevos pedidos.

- b. A partir de la tabla PEDIDOS:
  - i. El identificador del pedido, cliente, empleado y la fecha de pedido.
  - ii. Una vista que contenga todos los datos relaciones con el envío.

## **2.4. INSERTAR DATOS.**

1. Insertar 5 filas de datos de prueba en cada tabla.

## **2.5. CONSULTAS.**

1. Realizar las siguientes consultas:
  - a. Un listado del identificador y nombre de los clientes con su dirección completa y número de teléfono.
  - b. Un listado de los pedidos que ha realizado un cliente con un CliID determinado.
  - c. Un listado de los productos con su nombre, precio unitario, categoría y proveedor que se han pedido en un pedido determinado (conocido su DPedID).
  - d. Un listado de todos los productos, con los mismos datos anteriores de una categoría determinada y cuyo precio unitario sea superior a una cantidad dada (introducir los valores de la condición en función de los datos de prueba que se hayan insertado).



# PRÁCTICA 9. CREACIÓN DE APLICACIONES DE BASES DE DATOS CON BORLAND BUILDER C++

## 1. OBJETIVOS.

Al finalizar esta práctica el alumno será capaz de:

1. Importar una base de datos externa en SQL Server.
2. Crear y configurar un nuevo origen de datos (ODBC).
3. Crear aplicaciones sencillas que accedan a bases de datos utilizando Borland Builder.

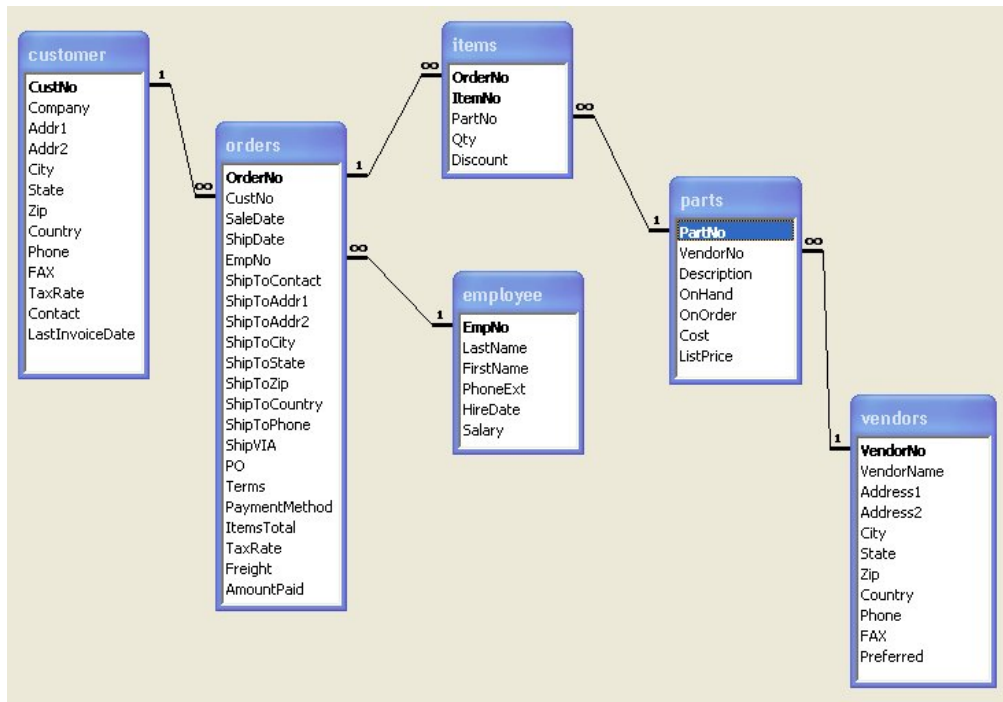
## 2. DETALLES.

### 2.1. IMPORTAR UNA BASE DE DATOS EXTERNA.

- a) Crear una nueva base de datos
  - Utilizando el Administrador Corporativo crear una base de datos llamada “Pedidos”.
- b) Importar datos
  - Pinchar sobre el nombre de la nueva base de datos.
  - En la parte derecha de la consola, en la pestaña *General*, y en el recuadro de *Base de datos*, seleccionar importar datos.

Los datos originales se crearon utilizando **Paradox 5.0**, y se encuentran en la ruta: C:\Archivos de programa\Archivos comunes\Borland Shared\Data.

Las tablas son las que se muestran a continuación:



- Poner el nombre de las seis tablas en mayúsculas.

c) Crear restricciones de clave ajena

- Indica a qué se debe el mensaje de error que aparece cuando se intenta establecer **EmpNo** en la tabla ORDERS como *foreign key* procedente de la tabla EMPLOYEE.
- Resolver el problema.

d) Añadir nuevos campos

- En la tabla EMPLOYEE añadir un campo llamado “Foto” de tipo *image*. En él se almacenará una fotografía del empleado.
- En la tabla EMPLOYEE añadir un campo llamado “Curriculum” de tipo *text*. En él se almacenará un archivo, en formato Word, con el currículum vitae del empleado.

## 2.2. CREAR UN NUEVO ORIGEN DE DATOS.

Abrir el administrador de orígenes de datos:

- Panel de Control, Herramientas Administrativas, Orígenes de datos (ODBC)

En la pestaña DNS de usuario:

- Agregar
- Seleccionar el controlador SQL Server
  - Nombre del origen de datos: **SQLPedidos**
- El resto a configurar por el alumno
  - Autenticación Windows NT
  - Establecer como predeterminada la base de datos *Pedidos*.
- Resto por defecto

- Idioma para los mensajes del sistema de SQL: Spanish
- Finalizar

## 2.3. CREAR UNA NUEVA APLICACIÓN.

- Abrir Borland Builder
- Crear una nueva aplicación
- Guardar el proyecto como *ApliPedidos*

## 2.4. AÑADIR COMPONENTES DE BASES DE DATOS.

1. Añadir un componente *Table*
  - En la paleta de componentes seleccionar la pestaña Data Access
  - Pinchar sobre el componente *Table* y luego sobre cualquier parte del *Form* abierto.

*Table* es un componente no visual, por lo que no importa dónde se ubique.
  - Cambiar el nombre de *Table1* por *Tabla1*.
  - En la propiedad *DatabaseName*, seleccionar **SQLPedidos**
2. Añadir una barra de estado
  - En la pestaña Win32

Hacer doble clic sobre el componente *Statusbar*. Esto añadirá una barra de estado en la parte inferior del formulario activo.
  - Poner la propiedad *AutoHint* a true.

## 2.5. COMPONENTES DE CONEXIÓN A LA BASE DE DATOS.

1. Añadir un componente *DataSource*
  - En la paleta de componentes seleccionar la pestaña Data Access
  - Pinchar sobre el componente *DataSource* y luego sobre cualquier parte del *Form* abierto.

Como es un componente no visual, no importa dónde se coloque.
  - Poner la propiedad *DataSet* a *Tabla1* (el nombre del componente *Table* añadido anteriormente).
2. Añadir un componente *DBGrid*
  - En la paleta de componentes seleccionar la pestaña Data Controls
  - Pinchar sobre el componente *DBGrid* y luego situarlo, por encima de la barra de estado, en la parte inferior de la ventana. Expandirlo tirando de su esquina derecha superior.
  - Poner la propiedad *DataSource* del *DBGrid* a *DataSource1* (el nombre del componente *DataSource* añadido anteriormente).
3. Visualizar el contenido de una tabla concreta en el *DBGrid*
  - Selecciona el componente *Tabla1*.
    - Poner la propiedad *TableName* a ORDERS.

Si te pide un login y un password, utiliza el del inicio de sesión (alumno, dieciseis).
    - Poner la propiedad *Active* a true.

La rejilla se rellenará con los datos de la tabla seleccionada.  
Si no es así... algo va mal.

4. Compila el proyecto (F9) y observa que los campos de la rejilla son editables.

## 2.6. AÑADIENDO ACCIONES ESTÁNDAR.

1. Añadir un componente *ImageList*.
  - En la paleta de componentes seleccionar la pestaña Win32
  - Pinchar sobre el componente *ImageList* y luego sobre cualquier parte del *Form* abierto.

Como es un componente no visual, no importa dónde se coloque.  
Contendrá iconos que representan acciones estándar como Cortar y Pegar.
2. Añadir un componente *ActionList*.
  - En la paleta de componentes seleccionar la pestaña Standard
  - Pinchar sobre el componente *ActionList* y luego sobre cualquier parte del *Form* abierto.

Como es un componente no visual, no importa dónde se coloque.
  - Poner la propiedad *Image* a *ImageList1* (el nombre del componente *ImageList* añadido anteriormente).
  - Hacer doble clic en el componente para visualizar el editor de acciones.
  - Hacer clic con el botón derecho sobre el editor del *ActionList* y elegir *NewStandard Action*.

Seleccionar las siguientes acciones: *TDataSetFirst*, *TDataSetLast*, *TDataSetNext*, *TDataSetPrior*, *TEditCopy*, *TEditCut* y *TEditPaste* (multiple selección). Pulsar O.K.

## 2.7. AÑADIENDO UN MENÚ.

1. Añadir un componente *MainMenu*.
  - En la paleta de componentes seleccionar la pestaña Standard
  - Pinchar sobre el componente *MainMenu* y luego sobre cualquier parte del *Form* abierto.

Como es un componente no visual, no importa dónde se coloque.
  - Poner la propiedad *Images* a *ImageList1* (el nombre del componente *ImageList* añadido anteriormente).
  - Hacer doble clic en el componente para visualizar el diseñador de menús.
    - Escribir &Archivo en la propiedad *Caption* del primer ítem del menú.

Dentro de Archivo, crear los elementos:

      - &Guardar.



- – (Un guión para poner un separador).
    - &Salir.
  - En el segundo elemento del nivel superior, poner &Editar en la propiedad *Caption*.
    - Del primer subelemento, poner su propiedad *Action* a *EditCut1*. Cambia el *Caption* a Cor&tar.
    - Poner la propiedad *Action* del siguiente subelemento a *EditCopy1*. Cambia el *Caption* a &Copiar.
    - Poner el siguiente a *EditPaste1*. Cambia el *Caption* a &Pegar.
  - En el tercer elemento del nivel superior, poner &Registro en la propiedad *Caption*.
    - Del primer subelemento, poner su propiedad *Action* a *DataSetFirst1*. Cambia el *Caption*.
    - Poner la propiedad *Action* del siguiente subelemento a *DataSetPrior1*. Cambia el *Caption*.
    - Poner el siguiente a *DataSetNext1*. Cambia el *Caption*.
    - Poner el último a *DataSetLast1*. Cambia el *Caption*.
2. Compila el programa y observa cómo funciona.

## 2.8. AÑADIENDO UNA BARRA DE HERRAMIENTAS.

1. Añadir un componente *ToolBar*.
  - ▶ En la paleta de componentes seleccionar la pestaña Win32
  - ▶ Hacer doble clic sobre el componente *ToolBar* para añadirlo al formulario.
  - ▶ Poner su propiedad *Images* a *ImageList1*.
  - ▶ Poner *ShowHint* a true.
2. Añadir botones a la barra de herramientas.
  - ▶ Seleccionar la barra de herramientas, hacer clic derecho y elegir *NewButton* tres veces.
  - ▶ Elegir después *NewSeparator*.
  - ▶ Añadir cuatro nuevos botones más.
3. Asignar acciones al primer conjunto de botones.
  - ▶ Seleccionar la propiedad *Action* de los tres primeros botones y asignarles *EditCut1*, *EditCopy1* y *EditPaste1*.
4. Asignar acciones al segundo conjunto de botones.
  - ▶ Seleccionar la propiedad *Action* de los cuatro segundos botones y asignarles *DataSetFirst1*, *DataSetPrior1*, *DataSetNext1*, *DataSetLast1*.
5. Compila el programa y observa cómo funciona.

The screenshot shows a Visual Basic form titled 'Form1'. It contains several controls:

- Form Fields:**
  - Numero Empleado:** A text box containing the value '2'.
  - Inicio Contrato:** A date picker showing '28/12/1988'.
  - Salario:** A text box containing the value '40000'.
  - (Blob):** A label next to a large empty text area.
  - Aemo):** A label next to another large empty text area.
- Order List:** A table with columns: Pedido, Item, Part, Desc., and Cantid. It contains one row with data: 1003, 1, 1313, 0, 5.
- Order Details Table:** A table with columns: OrderNo, CustNo, SaleDate, ShipDate, EmpNo, ItemsTotal, TaxRate, Freight, and Amo. It contains six rows of data.

| Pedido | Item | Part | Desc. | Cantid |
|--------|------|------|-------|--------|
| 1003   | 1    | 1313 | 0     | 5      |

| OrderNo | CustNo | SaleDate   | ShipDate            | EmpNo | ItemsTotal | TaxRate | Freight | Amo |
|---------|--------|------------|---------------------|-------|------------|---------|---------|-----|
| 1003    | 1351   | 12/04/1988 | 03/05/1988 12:00:00 | 114   | 1250       | 4,5     | 0       |     |
| 1004    | 2156   | 17/04/1988 | 18/04/1988          | 145   | 7885       | 0       | 0       |     |
| 1005    | 1356   | 20/04/1988 | 21/01/1988 12:00:00 | 110   | 4807       | 0       | 0       |     |
| 1006    | 1380   | 06/11/1994 | 07/11/1988 12:00:00 | 46    | 31987      | 0       | 0       |     |
| 1007    | 1384   | 01/05/1988 | 02/05/1988          | 45    | 6500       | 0       | 0       |     |
| 1008    | 1510   | 03/05/1988 | 04/05/1988          | 12    | 1449,5     | 0       | 0       |     |

## 2.9. AÑADIENDO CONTROLES PARA LA TABLA EMPLOYEE.

- De manera similar a lo realizado para la tabla ORDERS, añade los componentes necesarios para tener acceso a la tabla EMPLOYEE. Estos componentes servirán para visualizar información de los empleados. No es necesario añadir un DBGrid.
- Coloca los componentes dentro de un panel.
- Añade un control DBNavigator para poder acceder a los distintos campos.

## 2.10. AÑADIENDO CONTROLES PARA LA TABLA ITEMS.

- Utiliza un componente DBCtrlGrid para acceder a la tabla ITEMS.
- Indica cuál es la diferencia con un DBGrid.
- Investiga cómo se utiliza y realiza el diseño que aparece en la figura.

## TIPOS DE DATOS. M. SQL SERVER 7.0

|          |   |  |
|----------|---|--|
| Binarios | <b>binary</b> [( <i>n</i> )]<br>Datos binarios de longitud fija de <i>n</i> bytes. El argumento <i>n</i> debe ser un valor comprendido entre 1 y 8.000. El tamaño de almacenamiento es <i>n</i> +4 bytes.   | Utilice <b>binary</b> cuando prevea que las entradas de datos de una columna van a mantener aproximadamente el mismo tamaño.   |
|          | <b>varbinary</b> [( <i>n</i> )]<br>Datos binarios de longitud variable de <i>n</i> bytes. El argumento <i>n</i> debe ser un valor comprendido entre 1 y 8.000. El tamaño de almacenamiento es la longitud actual de los datos escritos + 4 bytes, no <i>n</i> bytes. Los datos escritos pueden tener una longitud de 0 caracteres. El sinónimo de SQL-92 para <b>varbinary</b> es <b>binary varying</b> . | Utilice <b>varbinary</b> cuando prevea que el tamaño de las entradas de datos de una columna va a variar considerablemente.  |
|          | <b>image</b><br>Datos binarios de longitud variable, desde 0 hasta $2^{31}-1$ (2.147.483.647) bytes.  | Los datos de tipo <b>image</b> se almacenan como una cadena de bits y no son interpretados por SQL Server. Cualquier interpretación de los datos de una columna <b>image</b> debe ser realizada por la aplicación. Por ejemplo, una aplicación podría almacenar datos en una columna <b>image</b> con el formato BMP, TIFF, GIF o JPEG. Depende de la aplicación que lee los datos de la columna <b>image</b> , reconocer o no el formato de los datos y mostrarlos correctamente. Todo lo que hace una columna <b>image</b> es proporcionar una ubicación para almacenar la secuencia de bits que conforman los datos de la imagen. |

|          |  |  |
|----------|--|--|
| Carácter | <p><b>char[(n)]</b></p> <p>Datos de caracteres no Unicode de longitud fija, con <i>n</i> caracteres. <i>n</i> tiene que estar comprendido entre 1 y 8.000. El tamaño de almacenamiento es <i>n</i> bytes. El sinónimo en SQL-92 para <b>char</b> es <b>character</b>.</p>  | <p>Utilice <b>char</b> cuando prevea que las entradas de datos de una columna van a mantener sistemáticamente el mismo tamaño.</p> <p>Si los datos que se van a almacenar tienen una longitud mayor que el número de caracteres permitido, se truncan los datos. Por ejemplo, si una columna se define como <b>char(10)</b> y en la columna se almacena el valor “Esta es una cadena de caracteres realmente larga”, Microsoft® SQL Server™ trunca la cadena de caracteres a “Esta es un”.</p> <p>El tipo de datos <b>char</b> es un tipo de datos de longitud fija cuando se especifica la cláusula NOT NULL. Si en una columna <b>char</b> NOT NULL se inserta un valor más corto que la longitud de la columna, el valor se rellena a la derecha con blancos hasta completar el tamaño de la columna. Por ejemplo, si una columna se define como <b>char(10)</b> y el dato que se va a almacenar es “música”, SQL Server almacena este dato como “música___”, donde “_” indica un blanco.</p> |
|          | <p><b>varchar[(n)]</b></p> <p>Datos de caracteres no Unicode de longitud variable con una longitud máxima de <i>n</i> caracteres. <i>n</i> tiene que ser un valor comprendido entre 1 y 8.000. El tamaño de almacenamiento es la longitud actual de los datos, no <i>n</i> bytes. Los datos especificados pueden tener una longitud de 0 caracteres. Los sinónimos en SQL-92 para <b>varchar</b> son <b>char varying</b> o <b>character varying</b>.</p> | <p>Utilice <b>varchar</b> cuando prevea que el tamaño de las entradas de datos de una columna va a variar considerablemente.</p> <p>El tipo de datos <b>varchar</b> es de longitud variable. Los valores más cortos que el tamaño de la columna no se rellenan a la derecha hasta completar el tamaño de la misma. Si la opción ANSI_PADDING estaba desactivada cuando se creó la columna, cualquier blanco a la derecha será recortado de los valores de carácter almacenados en la columna. Si ANSI_PADDING estaba activado cuando se creó la columna, los blancos a la derecha no se recortarán</p>   |
|          | <p><b>text</b></p> <p>Datos no Unicode de longitud variable, de la página de códigos del servidor y con una longitud máxima de 2<sup>31</sup>-1 (2.147.483.647) caracteres. Cuando la página de códigos del servidor utiliza caracteres de doble byte, el almacenamiento sigue siendo de 2.147.483.647 bytes. Dependiendo de la cadena de caracteres, el espacio de almacenamiento puede ser inferior a 2.147.483.647 bytes.</p>                         |  |

|         |   |   |
|---------|---|---|
| Unicode | <p><b>nchar(<i>n</i>)</b></p> <p>Datos de carácter Unicode de longitud fija, con <i>n</i> caracteres. <i>n</i> debe estar comprendido entre 1 y 4.000. El tamaño de almacenamiento es dos veces <i>n</i> bytes. Los sinónimos en SQL-92 para <b>nchar</b> son <b>national char</b> y <b>national character</b>.</p>   | <p>Utilice <b>nchar</b> cuando prevea que las entradas de datos de una columna van a mantener aproximadamente el mismo tamaño.</p>  |
|         | <p><b>nvarchar(<i>n</i>)</b></p> <p>Datos de carácter Unicode de longitud variable, con <i>n</i> caracteres. <i>n</i> debe ser un valor comprendido entre 1 y 4.000. El tamaño de almacenamiento, en bytes, es dos veces el número de caracteres especificados. Los datos especificados pueden tener una longitud de 0 caracteres. Los sinónimos en SQL-92 para <b>nvarchar</b> son <b>national char varying</b> y <b>national character varying</b>.</p> | <p>Utilice <b>nvarchar</b> cuando prevea que el tamaño de las entradas de datos de una columna va a variar considerablemente.</p>   |
|         | <p><b>ntext</b></p> <p>Datos Unicode de longitud variable con una longitud máxima de <math>2^{30} - 1</math> (1.073.741.823) caracteres. El almacenamiento, en bytes, es dos veces el número de caracteres especificado. El sinónimo en SQL-92 para <b>ntext</b> es <b>national text</b>.</p>   | <p>La especificación Unicode define un esquema de codificación único para prácticamente todos los caracteres usados con más frecuencia en todo el mundo. Todos los equipos convierten de forma coherente los patrones de bits de los datos Unicode en caracteres con la especificación única de Unicode. Esto asegura que el mismo patrón de bits se convierte siempre al mismo carácter en todos los equipos. Los datos se pueden transferir libremente desde una base de datos o equipo a otro, sin preocuparse de que el sistema que los reciba pueda convertir los patrones de bits de forma correcta</p> |

|                 |   |  |
|-----------------|---|--|
| De fecha y hora | <p><b>datetime</b></p> <p>Datos de fecha y hora comprendidos entre el 1 de enero de 1753 y el 31 de diciembre de 9999, con una precisión de un trescientosavo de segundo, o 3,33 milisegundos. Los valores se redondean con incrementos de 0,000, 0,003 o 0,007 milisegundos, como se muestra en la tabla</p> <p>Microsoft® SQL Server™ almacena internamente los valores de tipo de datos <b>datetime</b> como enteros de 4 bytes. Los primeros 4 bytes almacenan el número de días antes o después de la <i>fecha base</i>, el 1 de enero de 1900. La fecha base es la fecha de referencia del sistema. Los valores para <b>datetime</b> anteriores al 1 de enero de 1753 no se permiten. Los otros 4 bytes almacenan la hora del día representada como el número de milisegundos después de media noche.</p> | <p>Utilice <b>datetime</b> para almacenar datos del intervalo que va desde el 1 de enero de 1753 hasta el 31 de diciembre del 9999 (para cada valor se necesitan 8 bytes de espacio de almacenamiento).</p>        |
|                 | <p><b>smalldatetime</b></p> <p>Datos de fecha y hora desde el 1 de enero de 1900 al 6 de junio de 2079, con precisión de minutos.</p> <p>El tipo de datos <b>smalldatetime</b> almacena las fechas y horas del día con menor precisión que <b>datetime</b>. SQL Server almacena los valores <b>smalldatetime</b> como enteros de 2 bytes. Los dos primeros bytes almacenan el número de días después del 1 de enero de 1900. Los otros dos, almacenan el número de minutos desde media noche. El intervalo de datos es del 1 de enero de 1900 al 6 de junio de 2079, con precisión de minutos.</p>  | <p>Utilice <b>smalldatetime</b> para almacenar fechas en el intervalo que va desde el 1 de enero de 1900 hasta el 6 de junio del año 2079 (para cada valor se necesitan 4 bytes de espacio de almacenamiento).</p> |

|  |   |  |   |                          |                          |      |          |         |       |           |         |   |
|--|---|--|---|--------------------------|--------------------------|------|----------|---------|-------|-----------|---------|---|
| Numéricos  | Enteros   | <b>int</b><br>Datos enteros (números enteros) comprendidos entre $-2^{31}$ (-2.147.483.648) y $2^{31} - 1$ (2.147.483.647). El tamaño de almacenamiento es 4 bytes. El sinónimo en SQL-92 para <b>int</b> es <b>integer</b> .  | Los objetos y expresiones enteras se pueden usar en cualquier operación matemática. Cualquier fracción generada por estas operaciones será truncada, no redondeada. Por ejemplo, SELECT 5/3 devuelve el valor 1, no el valor 2 que devolvería si se redondeara el resultado fraccionario. |                          |                          |      |          |         |       |           |         |   |
|  |   | <b>smallint</b><br>Datos enteros comprendidos entre $-2^{15}$ (-32.768) y $2^{15} - 1$ (32.767). El tamaño de almacenamiento es 2 bytes.   | Los tipos de datos enteros son los únicos que se pueden usar con la propiedad IDENTITY, que es un número que se incrementa automáticamente. La propiedad IDENTITY se usa normalmente para generar automáticamente números exclusivos de identificación o claves principales.              |                          |                          |      |          |         |       |           |         |   |
|  |   | <b>tinyint</b><br>Datos enteros comprendidos entre 0 y 255. El tamaño de almacenamiento es de 1 byte.  | No es necesario incluir los datos enteros entre comillas simples como los datos de carácter o de fecha y hora.  |                          |                          |      |          |         |       |           |         |   |
|  | Decimales   | <b>decimal</b> [( <i>p</i> , <i>s</i> )] y <b>numeric</b> [( <i>p</i> , <i>s</i> )]<br>Números de precisión y escala fijas. Cuando se utiliza la precisión máxima, los valores permitidos están comprendidos entre $-10^{38} - 1$ y $10^{38} - 1$ . Los sinónimos de SQL-92 para <b>decimal</b> son <b>dec</b> y <b>dec</b> ( <i>p</i> , <i>s</i> ).   | Use el tipo de datos <b>decimal</b> para almacenar números con decimales cuando los valores de datos se deban almacenar exactamente como se especifican.  |                          |                          |      |          |         |       |           |         |   |
|  | Numéricos aprox.  | <b>float</b> [( <i>n</i> )]<br>Es un número en coma flotante con un valor comprendido entre -1,79E+308 y 1,79E+308. <i>n</i> es el número de bits que se utilizan para almacenar la mantisa del número <b>float</b> en notación científica y por tanto dicta su precisión y el tamaño de almacenamiento. <i>n</i> tiene que ser un valor entre 1 y 53.<br><table><tr><td><i>n</i> es</td><td>Precisión</td><td>Tamaño de almacenamiento</td></tr><tr><td>1-24</td><td>7 cifras</td><td>4 bytes</td></tr><tr><td>25-53</td><td>15 cifras</td><td>8 bytes</td></tr></table><br>El tipo de datos <b>float</b> [( <i>n</i> )] de Microsoft® SQL Server™ cumple el estándar SQL-92 para todos los valores de <i>n</i> comprendidos entre 1 y 53. El sinónimo de <b>double precision</b> es <b>float</b> (53). | <i>n</i> es   | Precisión                | Tamaño de almacenamiento | 1-24 | 7 cifras | 4 bytes | 25-53 | 15 cifras | 8 bytes | Los tipos de datos numéricos aproximados no almacenan los valores exactos especificados para muchos números; almacenan una aproximación muy precisa del valor. Para muchas aplicaciones, la pequeña diferencia entre el valor especificado y la aproximación almacenada no es apreciable. Sin embargo, a veces la diferencia se hace notar. Debido a esta naturaleza aproximada de los tipos de datos <b>float</b> y <b>real</b> , no los use cuando necesite un comportamiento numérico exacto, como, por ejemplo, en aplicaciones financieras, en operaciones que conlleven un redondeo o en comprobaciones de igualdad. En su lugar, use los tipos de datos enteros, <b>decimal</b> , <b>money</b> o <b>smallmoney</b> . |
|  |   | <i>n</i> es  | Precisión   | Tamaño de almacenamiento |                          |      |          |         |       |           |         |   |
| 1-24   | 7 cifras  | 4 bytes  |   |                          |                          |      |          |         |       |           |         |   |
| 25-53  | 15 cifras   | 8 bytes  |   |                          |                          |      |          |         |       |           |         |   |
| <b>real</b><br>Datos numéricos en coma flotante entre $-3.40E + 38$ y $3.40E + 38$ . El tamaño de almacenamiento es 4 bytes. En SQL Server, el sinónimo de <b>real</b> es <b>float</b> (24). | Evite usar columnas <b>float</b> o <b>real</b> en las condiciones de búsqueda de la cláusula WHERE, especialmente los operadores = y <>. Es mejor limitar las columnas <b>float</b> y <b>real</b> a las comparaciones > o <.<br>La especificación IEEE 754 proporciona cuatro modos de redondeo: redondear al más cercano, redondear hacia arriba, redondear hacia abajo y redondear hacia cero. Microsoft® SQL Server™ usa el redondeo hacia arriba. |  |   |                          |                          |      |          |         |       |           |         |   |

|        |  |   |
|--------|--|---|
| Moneda | <b>money</b><br>Valores de moneda comprendidos entre $-2^{63}$ (-922.337.203.685.477,5808) y $2^{63} - 1$ (+922.337.203.685.477,5807), con una precisión de una diezmilésima de la unidad monetaria. El tamaño de almacenamiento es 8 bytes. | <p>No es necesario incluir los datos de moneda o monetarios entre comillas simples. Sin embargo, el valor de los datos de moneda debe ir precedido del símbolo de moneda adecuado. Por ejemplo, para especificar 100 libras inglesas, use £100.</p> <p>Los tipos de datos <b>money</b> y <b>smallmoney</b> están limitados a cuatro espacios decimales. Use el tipo de datos <b>decimal</b> si se necesitan más espacios decimales.</p> <p>Use un punto para separar las unidades parciales de moneda, como céntimos, de las unidades completas de moneda. Por ejemplo, 2.15 puede especificar 2 dólares y 15 centavos.</p> <p>En las constantes <b>money</b> o <b>smallmoney</b> no se permiten los separadores de coma, aunque el formato de presentación de estos tipos de datos los incluye. Sólo puede especificar separadores de coma en las cadenas de caracteres convertidas explícitamente a <b>money</b> o <b>smallmoney</b>.</p> |
|        | <b>smallmoney</b><br>Valores monetarios comprendidos entre -214.748,3648 y +214.748,3647, con una precisión de una diezmilésima de la unidad monetaria. El tamaño de almacenamiento es 4 bytes.  |   |

|            |  |
|------------|--|
| Especiales | <b>timestamp</b><br>Se utiliza para indicar la secuencia de actividades de SQL Server en una fila, representada como un número creciente en formato binario. Cuando se modifica una fila de una tabla, el valor de timestamp (la marca de tiempo) se actualiza con el valor actual de timestamp de la base de datos que se obtiene con la función @@DBTS. Los datos de tipo <b>timestamp</b> no están relacionados con la fecha ni la hora de una inserción o de un cambio en los datos. Si desea registrar automáticamente en qué momento se producen modificaciones en una tabla, utilice un tipo de datos <b>datetime</b> o <b>smalldatetime</b> para registrar los sucesos y los desencadenadores. |
|            | <b>bit</b><br>Puede ser un 1 o un 0. Utilice el tipo de datos <b>bit</b> para representar los valores TRUE (verdadero) o FALSE (falso), o SÍ o NO. Por ejemplo, un cuestionario para los clientes en el que se pregunte si una visita es la primera del cliente puede almacenarse en una columna de tipo <b>bit</b> .  |
|            | <b>uniqueidentifier</b><br>Se trata de un número hexadecimal de 16 bytes que hace referencia a un identificador exclusivo global (GUID). El GUID es especialmente útil cuando una fila debe ser única entre otras muchas. Por ejemplo, utilice el tipo de datos <b>uniqueidentifier</b> en una columna con números de identificación de los clientes para compilar una lista de clientes de una compañía en varios   |



|            |  |   |
|------------|--|---|
| De usuario | <p>Los tipos de datos <b>definidos por el usuario</b> se basan en los tipos de datos del sistema de Microsoft® SQL Server™. Se pueden utilizar cuando varias tablas deben almacenar el mismo tipo de datos en una columna y desea asegurarse de que dichas columnas tienen exactamente el mismo tipo de datos, longitud y condición de la aceptación de valores NULL. Por ejemplo, es posible crear un tipo de datos definido por el usuario con el nombre <b>códigoPostal</b> a partir del tipo de datos <b>char</b>.</p> | <p>Cuando cree un tipo de datos definido por el usuario, deberá suministrar estos parámetros:</p> <p>Nombre.</p> <p>Datos del sistema en el que se basa el nuevo tipo de datos.</p> <p>Aceptación de NULL (si el tipo de datos permite valores NULL).</p> <p>Cuando no está definida explícitamente la aceptación de valores NULL, se asignará en función de la configuración de los valores NULL ANSI predeterminada para la base de datos o para la conexión.</p> |
|------------|--|---|



## 1. ESPECIFICAR UN TIPO DE DATOS PARA UNA COLUMNA.

La asignación de un tipo de datos para cada columna es uno de los primeros pasos que deberá seguir para diseñar una tabla. Los tipos de datos definen qué valores están permitidos en cada columna. Para asignar un tipo de datos a una columna, puede utilizar tipos de datos de Microsoft® SQL Server™ o crear sus propios tipos de datos a partir de los del sistema. Por ejemplo, si sólo desea incluir nombres en una columna, puede asignar un tipo de datos de carácter para la misma. Asimismo, si desea que una columna sólo contenga números, puede asignar un tipo de datos numérico.

## 2. EXIGIR LA INTEGRIDAD DE LOS DATOS.

Los tipos de datos del sistema y los definidos por el usuario pueden utilizarse para exigir la integridad de los datos, ya que los datos que se agregan o que se modifican deben ajustarse al tipo especificado en la instrucción original CREATE TABLE. Por ejemplo, no podrá almacenar un nombre en una columna definida con un tipo de datos **datetime**, porque una columna **datetime** sólo acepta fechas válidas. Por lo general, mantenga datos de tipo numérico en las columnas numéricas, sobre todo si deben realizarse cálculos con ellos más adelante.

### 2.1. DATOS BINARIOS.

Los datos binarios se componen de números hexadecimales. Por ejemplo, el número decimal 245 corresponde al hexadecimal F5. Los datos binarios se almacenan mediante los tipos de datos **binary**, **varbinary** e **image** en Microsoft® SQL Server™. Una columna a la que se le asigne el tipo de datos **binary** debe tener la misma longitud fija (hasta 8 KB) para cada fila. En una columna a la que se le asigne el tipo de datos **varbinary**, las entradas pueden variar por lo que respecta al número de dígitos hexadecimales (hasta 8 KB) que contiene. Las columnas con datos **image** se pueden utilizar para almacenar datos binarios de longitud variable que superen los 8 KB como, por ejemplo, documentos de Microsoft Word, hojas de cálculo de Microsoft Excel e imágenes, como mapas de bits, archivos GIF (Graphics Interchange Format) y JPEG (Joint Photographic Experts Group).

En general, utilice **varbinary** para almacenar datos binarios, excepto si su longitud supera los 8 KB, en cuyo caso deberá utilizar el tipo de datos **image**. Es recomendable que la longitud definida de una columna binaria no supere la longitud máxima prevista para los datos binarios que deben almacenarse.

#### Consulte también:

**binary** y **varbinary**  
Utilizar datos binarios  
**image**

Utilizar tipos de datos  
Utilizar datos **text** e **image**

## 2.2. DATOS DE CARÁCTER.

Se define como dato de carácter o alfanuméricos a cualquier combinación de letras, símbolos y caracteres numéricos. Por ejemplo, son datos de carácter válidos “928”, “Jiménez” y “(0\*&(%B99nh jkJ.”. En Microsoft® SQL Server™, los datos alfanuméricos se almacenan mediante los tipos de datos **char**, **varchar** y **text**. Utilice **varchar** cuando varíe el número de caracteres de las entradas de una columna, siempre que no haya ninguna entrada que tenga una longitud mayor de 8 KB. Utilice **char** cuando todas las entradas de una columna tengan la misma longitud fija (hasta 8 KB). Las columnas de datos **text** pueden utilizarse para almacenar caracteres ASCII mayores de 8 KB. Por ejemplo, dado que los documentos HTML son caracteres ASCII y suelen ocupar más de 8 KB, se pueden almacenar en columnas **text** en SQL Server antes de verlos con un explorador.

Se recomienda que la longitud definida para una columna de datos alfanuméricos no supere a la longitud máxima prevista para los datos alfanuméricos que vayan a almacenarse.

Para almacenar datos de caracteres internacionales en SQL Server, utilice los tipos de datos **nchar**, **nvarchar** y **ntext**.

### Consulte también:

|   |   |
|---|---|
| <b>char</b> y <b>varchar</b>                | Utilizar tipos de datos                   |
| <b>ntext</b> , <b>text</b> e <b>image</b>   | Utilizar datos <b>text</b> e <b>image</b> |
| Utilizar datos <b>char</b> y <b>varchar</b> | Utilizar datos Unicode                    |

## 2.3. DATOS UNICODE.

En Microsoft® SQL Server™, los tipos de datos tradicionales (no Unicode) permiten utilizar caracteres definidos por un juego de caracteres determinado. Durante la instalación de SQL Server, se elige un juego de caracteres que no se modifica mientras se mantiene la instalación. Si se utilizan los tipos de datos Unicode, una columna puede almacenar cualquier carácter que defina el estándar Unicode, lo que incluye a todos los caracteres definidos en los distintos juegos de caracteres. Los tipos de datos Unicode ocupan el doble de espacio que los que no lo son.

Se almacenan mediante los tipos de datos **nchar**, **nvarchar** y **ntext** de SQL Server. Utilice estos tipos de datos para las columnas que almacenen caracteres de más de un juego de caracteres. Utilice **nvarchar** cuando las entradas de una columna varíen en cuanto al número de caracteres Unicode que contienen (hasta 4.000). Utilice **nchar** cuando todas las entradas de una columna tengan la misma longitud fija (hasta 4.000 caracteres Unicode). Utilice **ntext** cuando alguna entrada de una columna ocupe más de 4.000 caracteres Unicode.

---

**Nota:** Los tipos de datos Unicode de SQL Server se basan en los tipos de datos National Character del conjunto de normas SQL-92. SQL-92 utiliza el carácter n como prefijo para identificar estos tipos de datos y valores.

**Consulte también:**

|   |                        |
|---|------------------------|
| <b>nchar</b> y <b>nvarchar</b>            | Estándar Unicode       |
| <b>ntext</b> , <b>text</b> e <b>image</b> | Utilizar datos Unicode |

**2.4. DATOS DE FECHA Y HORA.**

Los datos de fecha y hora constan de combinaciones válidas de fecha y hora. Por ejemplo, datos válidos de fecha y hora pueden ser “4/01/98 12:15:00:00:00 p.m.” y “1:28:29:15:01 a.m. 17/8/98”. Los datos de fecha y hora se almacenan mediante los tipos de datos **datetime** y **smalldatetime** de Microsoft® SQL Server™. Utilice **datetime** para almacenar datos del intervalo que va desde el 1 de enero de 1753 hasta el 31 de diciembre del 9999 (para cada valor se necesitan 8 bytes de espacio de almacenamiento). Utilice **smalldatetime** para almacenar fechas en el intervalo que va desde el 1 de enero de 1900 hasta el 6 de junio del año 2079 (para cada valor se necesitan 4 bytes de espacio de almacenamiento).

**Consulte también:**

|                                |  |
|--------------------------------|--|
| Utilizar tipos de datos        | <b>datetime</b> y <b>smalldatetime</b> |
| Utilizar datos de fecha y hora |  |

**2.5. DATOS NUMÉRICOS.**

Los datos numéricos se componen exclusivamente de números. Incluyen números positivos y negativos, decimales, fracciones y números enteros.

**2.5.1. Datos enteros.**

Son datos enteros los números enteros positivos y negativos, como -15, 0, 5 y 2.509. Los datos enteros se almacenan mediante los tipos de datos **int**, **smallint** y **tinyint** de Microsoft® SQL Server™. El tipo de datos **int** puede almacenar un intervalo mayor de enteros que **smallint**, que a su vez puede almacenar un intervalo mayor de números que **tinyint**. Utilice el tipo de datos **int** para almacenar números del intervalo comprendido entre -2.147.483.648 y 2.147.483.647 (para cada valor se necesitan 4 bytes de espacio de almacenamiento). Utilice el tipo de datos **smallint** para almacenar números del intervalo que va desde -32.768 hasta 32.767 (para cada valor se necesitan 2 bytes de espacio de almacenamiento), y el tipo de datos **tinyint** para almacenar números del intervalo que va desde 0 hasta 255 (para cada valor se necesita 1 byte de espacio de almacenamiento).

**2.5.2. Datos decimales.**

Los datos decimales se componen de datos de los que se almacena hasta el dígito menos significativo. Estos datos se almacenan mediante los tipos de datos **decimal** o **numeric** de SQL Server. El número de bytes necesario para almacenar un valor **decimal** o **numeric** depende del número total de dígitos de los datos y del número

de decimales que haya a la derecha del separador decimal. Por ejemplo, se necesitan más bytes para almacenar el valor 19.283,29383 que para almacenar el valor 1,1.

En SQL Server, el tipo de datos **numeric** es sinónimo del tipo de datos **decimal**.

### 2.5.3. Datos numéricos aproximados.

Los datos numéricos aproximados (en coma flotante) constan de datos que se almacenan con tanta precisión como permite el sistema de numeración binario. Los datos numéricos aproximados se almacenan mediante los tipos de datos **float** y **real** de SQL Server. Por ejemplo, dado que, en notación decimal, la fracción “un tercio” se expresa como 0,333333 (hasta el infinito), este valor no se puede representar con total precisión mediante los datos decimales aproximados. Por lo tanto, el valor que se recupera de SQL Server puede no coincidir exactamente con el que se almacenó inicialmente en la columna. Otros ejemplos de aproximaciones numéricas son los valores de coma flotante que acaban en 0,3, 0,6 y 0,7.

#### Consulte también:

|                                |   |
|--------------------------------|---|
| <b>decimal y numeric</b>       | Utilizar tipos de datos                     |
| <b>float y real</b>            | Utilizar datos enteros                      |
| <b>int, smallint y tinyint</b> | Utilizar datos <b>decimal, float y real</b> |

### 2.6. DATOS DE MONEDA.

Los datos de moneda representan cantidades positivas o negativas de dinero. Los datos de moneda se almacenan mediante los tipos de datos **money** y **smallmoney** de Microsoft® SQL Server™. Los datos de moneda pueden almacenarse con una precisión máxima de cuatro decimales. Utilice el tipo de datos **money** para almacenar valores del intervalo comprendido entre -922.337.203.685.477,5808 y +922.337.203.685.477,5807 (para almacenar cada valor se necesitan 8 bytes). Utilice el tipo de datos **smallmoney** para almacenar valores del intervalo comprendido entre -214.748,3648 y 214.748,3647 (para almacenar cada valor se necesitan 4 bytes). Si se necesita un número mayor de decimales, utilice el tipo de datos **decimal**.

#### Consulte también:

|                           |                          |
|---------------------------|--------------------------|
| <b>money y smallmoney</b> | Utilizar datos de moneda |
|                           | Utilizar tipos de datos  |

### 2.7. DATOS ESPECIALES.

Los datos especiales son aquellos que no corresponden a ninguna de las categorías de datos mencionadas anteriormente (datos binarios, de carácter, Unicode, de fecha y de hora, numéricos y de moneda).

Microsoft® SQL Server™ incluye cuatro tipos de datos especiales:

- **timestamp**  
Se utiliza para indicar la secuencia de actividades de SQL Server en una fila, representada como un número creciente en formato binario. Cuando se modifica una fila de una tabla, el valor de timestamp (la marca de tiempo) se actualiza con el valor actual de timestamp de la base de datos que se obtiene con la función @@DBTS. Los datos de tipo **timestamp** no están relacionados con la fecha ni la hora de una inserción o de un cambio en los datos. Si desea registrar automáticamente en qué momento se producen modificaciones en una tabla, utilice un tipo de datos **datetime** o **smalldatetime** para registrar los sucesos y los desencadenadores.
- **bit**  
Puede ser un 1 o un 0. Utilice el tipo de datos **bit** para representar los valores TRUE (verdadero) o FALSE (falso), o SÍ o NO. Por ejemplo, un cuestionario para los clientes en el que se pregunte si una visita es la primera del cliente puede almacenarse en una columna de tipo **bit**.
- **uniqueidentifier**  
Se trata de un número hexadecimal de 16 bytes que hace referencia a un identificador exclusivo global (GUID). El GUID es especialmente útil cuando una fila debe ser única entre otras muchas. Por ejemplo, utilice el tipo de datos **uniqueidentifier** en una columna con números de identificación de los clientes para compilar una lista de clientes de una compañía en varios países.
- definido por el usuario  
Permite que el usuario defina tipos de datos, como, por ejemplo, **códigoProducto**, que se basa en el tipo de datos **char** y que consta de dos letras mayúsculas seguidas de un número de proveedor de cinco cifras.

#### Consulte también:

|                         |  |
|-------------------------|--|
| <b>bit</b>              | Utilizar tipos de datos                |
| <b>timestamp</b>        | Utilizar datos especiales              |
| <b>uniqueidentifier</b> | Utilizar datos <b>Uniqueidentifier</b> |

### 3. CREAR TIPOS DE DATOS DEFINIDOS POR EL USUARIO.

Los tipos de datos definidos por el usuario se basan en los tipos de datos del sistema de Microsoft® SQL Server™. Se pueden utilizar cuando varias tablas deben almacenar el mismo tipo de datos en una columna y desea asegurarse de que dichas columnas tienen exactamente el mismo tipo de datos, longitud y condición de la aceptación de valores NULL. Por ejemplo, es posible crear un tipo de datos definido por el usuario con el nombre **códigoPostal** a partir del tipo de datos **char**.

Cuando cree un tipo de datos definido por el usuario, deberá suministrar estos parámetros:

- Nombre

- Datos del sistema en el que se basa el nuevo tipo de datos.
- Aceptación de NULL (si el tipo de datos permite valores NULL).

Cuando no está definida explícitamente la aceptación de valores NULL, se asignará en función de la configuración de los valores NULL ANSI predeterminada para la base de datos o para la conexión.

---

**Nota:** Si se crea un tipo de datos definido por el usuario en la base de datos **model**, existirá en todas las nuevas bases de datos definidas por el usuario. Sin embargo, si el tipo de datos se crea en una base de datos definida por el usuario, sólo existirá en esa base de datos.

---

**Consulte también:**

|              |                         |
|--------------|-------------------------|
| ALTER TABLE  | Utilizar tipos de datos |
| CREATE TABLE |                         |



## BIBLIOGRAFÍA



# BIBLIOGRAFÍA

## 1. BIBLIOGRAFÍA BÁSICA.

- [1] A. Silberschatz, H.F. Korth, y S. Sudarshan, *Fundamentos de bases de datos*, 4ª edición, McGraw Hil, 2002.

Libro muy completo y claro. Se ha utilizado como base en los temas 2 y 3 para explicar el modelado Entidad-Relación y el Relacional. Debido a su amplitud podría utilizarse como libro de texto no sólo de este curso de bases de datos, sino también de cursos posteriores donde se traten temas de bases de datos orientadas a objetos, XML, o consultas y procesamiento avanzado de transacciones, entre otros.

- [2] C. J. Date. *Introducción a los Sistemas de Bases de Datos*, 7ª edición. Prentice-Hall, 2001.

Un clásico, sin duda. Conciso y muy orientado a ejemplos concretos introduciendo el uso de SQL desde las primeras páginas. Con un lenguaje formal concede mucha importancia al álgebra relacional. Dentro de nuestro temario, útil para estudiar el tema 1 de una forma muy detallada. También puede utilizarse en los temas 3 y 4 para consolidar las ideas de restricciones y normalización.

- [3] R.A. Elmasri, y S.B. Navathe. *Fundamentos de los Sistemas de Bases de Datos*, 3ª edición. Prentice-Hal, 2002.

Por su contenido puede ser considerado como libro de texto básico. A nuestro juicio menos estructurado que el anterior aunque algunos temas los trata con mayor profundidad. Se ha utilizado en el tema 4, normalización, y puntualmente en los temas relacionados con modelado E/R y Relacional, transacciones y bloqueos.

- [4] G. Gardarin. *Dominar las bases de datos. Modelos y lenguajes*. Enrolles y Ediciones Gestión 2000, 1994.

Un libro que, como su título indica, se centra en los modelos de datos y en el lenguaje SQL. Se ha utilizado en el tema 1 para ofrecer una visión de los modelos en red y jerárquico. Ha sido el texto empleado como básico en el tema 6 sobre índices, debido a la clara información que presenta sobre métodos de acceso indexados y los métodos de acceso hash, por direccionamiento calculado.

- [5] G. W. Hansen y J.V. Hansen. *Diseño y Administración de Bases de Datos*, 2ª edición. Prentice Hall, 1997.

Con un estilo muy americano, al introducir los temas con casos concretos de empresas con sus problemas y soluciones, ofrece luz a los temas y explica de forma clara muchos conceptos. Se ha utilizado como texto básico para desarrollar el tema 4 sobre normalización y algunos aspectos del tema 3 sobre el modelo Relacional. Resulta interesante leer alguno de sus apartados para completar los conceptos básicos sobre bases de datos.

- [6] B.W. McEwan y D. Solomon. *Teach Yourself Transact-SQL in 21 days*. SAMS-Publishing, 1997.

Solo SQL y principalmente Transact-SQL. Este libro se puede utilizar como una guía para ir aprendiendo y practicando desde los conceptos básicos de SQL hasta algunos bastante avanzados. Aunque los ejemplos son con SQL suele indicarse lo que es ANSI SQL o específicamente Transact. Los ejercicios que va proponiendo se realizan sobre la base de datos de ejemplo *pubs* de Microsoft SQL Server. Se ha utilizado como base para desarrollar el tema 5 sobre SQL.

- [7] L. Grau e I. López. *Problemas de Bases de Datos*. Editorial Sanz y Torres S.L., 1998.

Libro de problemas conteniendo enunciados y ejercicios resueltos sobre el modelo Entidad-Relación, el Álgebra Relacional, el cálculo relacional, SQL, QBE y QUEL. Se ha utilizado únicamente la parte de los ejercicios del modelo Entidad Relación, resolviéndolos en nuestro caso utilizando el modelo extendido (el libro utiliza el básico) y obteniendo posteriormente el modelo relacional.

- [8] A. de Miguel, P. Martínez, E. Castro, J.M<sup>a</sup> Cavero, D. Cuadra, A.M<sup>a</sup>. Iglesias y C. Nieto. *Diseño de Bases de Datos. Problemas resueltos.*, Ra-Ma, 2001.

Excelente libro de problemas resueltos que proporciona un gran número de enunciados complejos que modeliza mediante E/R extendido para posteriormente ofrecer, de algunos, el Relacional. Ofrece además algunos problemas sobre Normalización, diseño de bases de datos distribuidas y diseño con herramientas CASE. Se ha utilizado en los problemas de la asignatura más complejos del modelo E/R. La solución adoptada coincide en su mayor parte con la propuesta en el libro, aunque el modelo Relacional obtenido en nuestra asignatura se expresa de manera diferente.

## 2. BIBLIOGRAFÍA COMPLEMENTARIA.

- [9] S. Bjeletich, G. Mable y otros. *Microsoft SQL Server 7, Al descubierto*. SAMS, 1999.

Exhaustivo libro sobre SQL Server 7, utilizado no sólo para las prácticas, sino también para explicar algunos temas de las clases teóricas, como son los aspectos relacionados con la Desnormalización, del tema 4. También puede utilizarse en la realización de trabajos de la asignatura al profundizar en diversos aspectos prácticos no vistos durante el curso.

- [10] M. J. Ramos, A. Ramos y F. Montero. *Desarrollo de aplicaciones en entornos de 4<sup>a</sup> generación y con herramientas CASE*. Mc Graw Hill, 2000.

Aunque presenta conceptos básicos sobre bases de datos la mayor virtud de este libro consiste en su aplicación directa mediante ejercicios a Oracle 8i. Explica en profundidad el PL/SQL y también introduce la utilización de Oracle Forms y Oracle Reports.

- [11] A. de Miguel y M. Piattini. *Concepción y Diseño de Bases de Datos. Del Modelo E/R al Modelo Relacional*. RA-MA, 1993.

Extenso libro que se centra principalmente en los aspectos de modelado. Permite completar ideas sobre los modelos CODASYL, en Red y Jerárquico. También explica con claridad el modelo Entidad/Relación extendido.

- [12] A. de Miguel y M. Piattini. *Fundamentos y modelos de Bases de Datos*. RA-MA, 1997.

Similar al anterior. Menos extenso y más concreto. Contiene más información sobre SQL y comenta aspectos de los Sistemas de Información y Seguridad.

- [13] P. Beynon-Davies. *Database Systems*, 2th edition, Macmillan Press Ltd., 2000.

Libro conciso y claro que permite adquirir conceptos generales así como realizar una primera aproximación a varios estándares comerciales de Sistemas Gestores de Bases de Datos, como son Access y Oracle. También proporciona una visión del SQL3 y de los sistemas Orientados a Objeto.

### 3. BIBLIOGRAFÍA AUXILIAR.

- [14] S. Bobrowski. *Oracle 8i para Windows NT. Edición de aprendizaje*. Oracle Press - Osborne McGraw Hill.2000.

- [15] S. Youness. *Professional Data Warehousing with SQL Server 7.0 and OLAP Services*. Wrox Press, 2000.

- [16] J.R. Groff y P. N Weinberg, *Aplique SQL*. Osborne McGraw Hill, 1997.

- [17] K. Henderson. *The Guru's Guide to Transact-SQL*. Addison Wesley, 2000.